10.601.066 09.08.2003

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 953 905 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
03.11.1999 Bulletin 1999/44

(51) Int. Cl.[6]: **G06F 9/46**

(21) Application number: **99303005.5**

(22) Date of filing: **19.04.1999**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **27.04.1998 JP 11750598**

(71) Applicant: **SONY CORPORATION**
Tokyo 141 (JP)

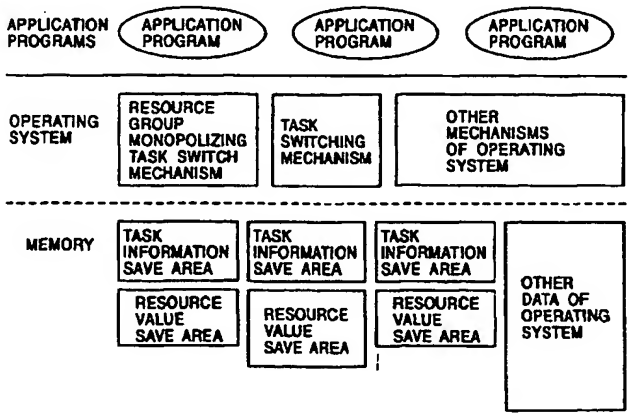
(72) Inventors:
- **Wakita, Yoshiro,**
  Intellectual Property Dpt.
  Shinagawa-ku, Tokyo 141 (JP)
- **Murata, Seiji,**
  Intellectual Property Dpt.
  Shinagawa-ku, Tokyo 141 (JP)
- **Gamo, Tsutomu,**
  Intellectual Property Dpt.
  Shinagawa-ku, Tokyo 141 (JP)

(74) Representative:
Robinson, Nigel Alexander Julian et al
D. Young & Co.,
21 New Fetter Lane
London EC4A 1DA (GB)

(54) Method and apparatus for switching tasks

(57) A multitasking system is arranged so that task switching can be performed efficiently in a short time. Resources usable by tasks are controlled in a state of being separated into a plurality of resource groups. When one of the tasks which has become active by task switching uses one of the resources, ascertainment is made as to whether the resource group to which the resource required by the task belongs is usable. If the resource group is usable by the active task, the active task is permitted to use the resource to advance its process. If the resource group is not usable by the task, the conditions relating to the resource group to which the required resource belongs are changed so that the task can use the resource group. Thereafter, the active task is permitted to use the resource to advance its process.

FIG. 3



EP 0 953 905 A2

Description

[0001] The present invention relates to a m thod of switching tasks in a multitasking system in which a plurality of tasks are executed in parallel with each other while independence of each task is ensured. The present invention also relates to a task switching apparatus for use in a multitasking system in which a plurality of tasks are executed in parallel with each other while independence of each task is ensured.

[0002] To enable task switching in conventional operating systems having a multitasking function, a task information save area for storing task information for controlling each of a plurality of tasks, and a resource value save area for saving resource values which designate the statuses of resources used for executing the task are maintained as a set of adjacent areas in a memory, as shown in Fig. 20. The resource values are, for example, values in registers for a central processing unit and values in registers for a floating-point arithmetic unit.

[0003] For example, when a task A is being executed, task switching is performed by stopping execution of the task A and starting execution of a task B. In this task switching, resource values reflecting the results of use of resources for the task A before switching are first saved in the memory together with task information about the task A. That is, when the task A is stopped, values representing the current statuses of the resources are saved as resource values in the memory.

[0004] Then, with respect to the task B to be next executed, resource values which were set as a result of the preceding execution of the task B using resources are read from the memory together with task information about the task B, and execution of the task B is thereafter started. As the resource values to be read from the memory, values indicating the statuses of the resources at the time when the task B was stopped have been set. When execution of the task B is started, the register values of the central processing unit, the register values of the floating-point arithmetic unit, etc., are restored, based on the resource values, to the statuses at the time when the task B was stopped. Thereafter, execution of the task B is restarted.

[0005] A central processing unit is known which has resource value saving hardware called "register file" used as an area for saving resources for each of a plurality of tasks, as shown in Fig. 21A. Another central processing unit is known which has resource value saving hardware called "register stack" and used as an area for saving resources for each of a plurality of tasks, as shown in Fig. 21B. In these central processing units, reads of resource values from a memory and writes of resource values to the memory at the time of task switching can be reduced, so that the task switching speed can be increased.

[0006] Conventionally, areas for saving task information and resourc values are maintained in a memory with respect to each of a plurality of tasks, as shown in Fig. 20. Task switching performed in the above-described manner, however, requires reading of resourc values from a memory and writing of resource values to the memory with respect to each of resources usable by tasks, and therefore entails problems described below.

(1) If there is a large number of resources (e.g., if a central processing unit has a multiplicity of registers), a need for reading and writing of a large amount of data by accessing a memory arises, so that the time required for task switching is long.

(2) If a resource is newly added, resource values of the added resource are also stored in the memory with respect to the tasks not using the added resource. The time required for task switching is thereby increased. For example, if a floating-point arithmetic unit is added for hardware function expansion, the tasks not using the function of the floating-point arithmetic unit also have register values of the floating-point arithmetic unit stored as resource values in the memory. As a result, the task switching time is increased.

(3) Even if the number of usable resources is increased, it is not always possible that each task will use all the resources. Ordinarily, some of the tasks use only a limited number of resources in the total number of usable resources. In the conventional task switching system, however, the resource values of all the usable resources are stored in the memory with respect to the tasks using only a limited number of resources in the total number of usable resources. Therefore, a considerably large amount of memory is consumed for task switching if the number of usable resources is increased.

If resource value saving hardware is prepared as shown in Fig. 21A or 21B, the problem (1) can be solved but the following is still a consideration.

(4) There is a limit to the capacity of available resource value saving hardware, and it is not always possible that prepared resource value saving hardware will suffice for any task switch. If resources are added or the number of switchable tasks is increased to such an extent that the prepared resource value saving hardware is insufficient, reading and writing performed by accessing the memory as shown in Fig. 20 are eventually required. Consequently, even if resource value saving hardware is provided to increase the task switching speed, the effect of increasing the speed is considerably low if resources are added or the number of switchable tasks is increased.

(5) A large-scale of resource value saving hardware is required with respect to the amount of resources to be handled. In integration of the desired central processing unit, etc., forming a semiconductor device, a considerably large semiconductor area is

occupied by the resource value saving hardware.

(6) If the amount of resources to be handled is increased, it is necessary to correspondingly increase the scale of resource value saving hardware or to reduce the number of tasks savable by resource value saving hardware.

In the case of an application of a task switching system such as shown in Fig. 20 to an apparatus requiring real-time processing, a problem described below arises especially when the amount of usable resources is increased.

(7) Since resource values of all usable resources are saved in the memory and read out from the memory each time task switching is performed, the time required for task switching is increased especially when the amount of usable resources is increased. In such a situation, processing, such as interrupt processing, which needs a high-speed start, cannot be performed by the task switching system. This problem is particularly considerable in the case of an apparatus requiring real-time processing.

In the case where task switching is performed as shown in Fig. 20, the memory is accessed to transfer a substantially large amount of data and the amount of data transferred by accessing the memory varies largely depending upon whether or not a cache memory or a register file can be used. Therefore, there is also a problem described below.

(8) The time required for starting a task by task switching varies largely and it is difficult to determine a processing timing when interrupt processing or the like is performed.

[0007] As described above, there are various problems relating to task switching.

[0008] In view of the above-described circumstances, an object of the present invention is to provide a task switching apparatus and method whereby the above-described problems of the conventional art can be reduced.

[0009] To achieve this object, according to the present one aspect of the present invention, there is provided a task switching apparatus in a multitasking system comprising a plurality of tasks executed in parallel with each other while independence of each task is ensured, each task having a resource value save area for its exclusive use, a plurality of resource groups for controlling resources in a separate state, a task switcher which switches the tasks so that a second one of the tasks becomes active in place of a first one of the tasks, a resource group usability checker for checking, when the active task uses one of the resources which belong to one of the plurality of groups, whether the active task has a right to use the resource group to which the resource to be used by the active task belongs, and a resource group monopolizing task switcher for giving the active task a right to use the resource group to which the resource to be used by the active task belongs if the active task has no right to use the resource group.

[0010] According to another aspect of th present invention, there is provided a task switching method in a multitasking system comprising an execution step of executing a plurality of tasks in parallel with each other while independence of each task is ensured, each task having a resource value save area for its exclusive use, a storing step of storing a plurality of resource groups in a separate state, a task switching step of switching the tasks so that a second one of the tasks becomes active in place of a first one of the tasks, a resource group usability checking step of checking, when the active task uses one of the resources which belong to one of the plurality of groups, whether the active task has a right to use the resource group to which the resource to be used by the active task belongs, and a resource group monopolising task switching step of giving the active task a right to use the resource group to which the resource to be used by the active task belongs if the active task has no right to use the resource group.

[0011] Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

Fig. 1 is a diagram showing the configuration of hardware used in a first embodiment of the present invention;

Fig. 2 a diagram for explaining resource groups and variables on a memory in the first embodiment of the present invention;

Fig. 3 is a diagram showing a state in which a plurality of tasks are executed in parallel with each other in accordance with one embodiment of the present invention.

Fig. 4 is a diagram showing a task information array in the first and second embodiment of the invention

Fig. 5 is a diagram showing a task information save area and a resource value save area in the first embodiment of the invention.

Fig. 6 is a flowchart of a task switching process in the first embodiment of the invention;

Fig. 7 is a flowchart of the process of a resource group usability check in the first embodiment of the invention;

Fig. 8 is a flowchart of a resource group monopolizing task switching process in the first embodiment of the invention;

Fig. 9 is a diagram showing the configuration of hardware registers in the second embodiment of the invention;

Fig. 10 is a diagram showing a task information save area and a resource value save area in the second embodiment of the invention;

Fig. 11 is a flowchart of a task switching process in which task switching is performed by using resource group use permission flags in the second

embodiment of the invention;

Fig. 12 is a flowchart of the process of a resource group usability check when task switching is performed by using resource group use permission flags in the second embodiment of the invention;

Fig. 13 is a flowchart of a resource group monopolizing task switching process when task switching is performed by using resource group use permission flags in the second embodiment of the invention;

Fig. 14 is a flowchart of a resource group saving process in the second embodiment of the invention;

Fig. 15 is a flowchart of a resource group reading process in the second embodiment of the invention;

Fig. 16 is a flowchart of a resource group save area securing process in the second embodiment of the invention;

Fig. 17 is a flowchart of a task switching process in which task switching is performed without using resource group use permission flags in the second embodiment of the invention;

Fig. 18 is a flowchart of the process of a resource group usability check when task switching is performed without using resource group use permission flags in the second embodiment of the invention;

Fig. 19 is a flowchart of a resource group monopolizing task switching process when task switching is performed without using resource group use permission flags in the second embodiment of the invention;

Fig. 20 a diagram showing a state in which a plurality of tasks are executed in parallel with each other in the conventional art;

Figs. 21A and 21B are diagrams for explaining high-seed task switching mechanisms in a conventional central processing unit;

Fig. 21A is a diagram showing a register file;

Fig. 21B is a diagram showing a register window; and

Fig. 22 is a diagram schematically showing the configuration of a television receiver which is an example of an application of the present invention.

1. Outline of Embodiments of the Present Invention

[0012] According to at least preferred embodiments of the present invention, resources, e.g. registers for a central processing unit and registers for a floating-point arithmetic unit used for tasks and having values to be saved independent of each other with respect to the tasks are controlled by being separated into a plurality of groups called resource groups.

[0013] According to at least preferred embodiments of the present invention, when an active task uses a resource, a check called "resource group usability check" is executed for determination as to whether a resource gr up to which the resource belongs is usable by the active task. According to at least preferred embodiments of the present invention, if the active task is monopolizing the resource group, it is determined that the resource group is usable by th task. If usability of the resource is recognized as a result of the resource group usability check, the process in which the task is executed is advanced, as is the corresponding conventional process not using the method of at least preferred embodiments of the present invention. If the resource is not usable, a process called "resource group monopolizing task switching process" is started.

[0014] In the resource group monopolizing task switching process, saving and reading of resource values and changing of the resource group monopolizing state are performed with respect to the switching-object resource group. More specifically, the task which is presently monopolizing the switching-object resource group is identified and, if valid values have been set in some resources which belong to the resource group by the task presently monopolizing the resource group, they are saved in a resource value save area for the task. If the active task (i.e., the task being about to use the above-mentioned resource) has, in a resource value save area, stored valid values corresponding to the resource group, those values are read. After these operations, switching from the task monopolizing the resource group to the active task is performed.

[0015] A method and an apparatus for switching tasks in accordance with at least preferred embodiments of the present invention can be realized as a portion of software such as an operating system and can be realized by using hardware to which the present invention is applied. In either case, to enable normal execution of a process including a resource group usability check and processing for resource group monopolizing task switching, some flags and variables are prepared and the values of them are set and referred to as desired.

[0016] A resource group usability check is a process performed considerably frequently. Therefore, it is preferred that the resource group usability check is executed by hardware processing based on a resource group usability checking mechanism in prepared hardware instead of software processing based on an operating system or the like, because a higher processing speed can be achieved thereby.

[0017] For example, such a resource group usability check is executed by using a use permission flag which is provided as a function of hardware to indicate permission/inhibition of use of each of functional blocks. Each functional block is formed of one or more resource groups which provide a common function. More specifically, a group of registers for an arithmetic unit is handled as a functional block, for example.

[0018] If a resource group usability check is performed by using use permission flags indicating permission/inhibition of use of such functional blocks, the use permission flag corresponding to some of the functional blocks containing an unusable resource group is set such as to indicate that use of the functional block con-

taining the unusable resource group is not permitted. In the resource group usability check, permission/inhibition of use of one of resource groups is recognized by referring to the corresponding use permission flag.

[0019] Alternatively, a resource group use p rmission flag indicating permission/inhibition of use of each of resource groups may be provided as a function of hardware, and a resource group usability check may be executed by using the resource group use permission flag. To enable a resource group usability check by using resource group use permission flags indicating permission/inhibition of use of resources, the resource group use permission flag corresponding to an unusable one of resource groups is previously set such as to indicate that use of the resource group is not permitted. In the resource group usability check, permission/inhibition of use of one of the resource groups is recognized by referring to the corresponding resource group use permission flag.

[0020] Also, a register in which resource group monopolizing task variables are set in one-to-one correspondence with resource groups, and a register in which an active task variable set to a value corresponding to an active task is stored may be provided in the form of hardware and a resource group usability check may be performed by using the resource group monopolizing task variables and the active task variable. In a resource group usability check using the resource group monopolizing task variables and the active task variable, permission/inhibition of use of each resource group is ascertained by comparing the corresponding resource group monopolizing task variable and the active task variable.

[0021] If registers for storing flags or variables used for a resource group usability check are provided in the form of hardware as described above, a resource group usability can be performed without reducing the task execution efficiency.

[0022] On the other hand, from the viewpoint of performing a resource group monopolizing task switching process in a shorter time, it is also preferred that the greatest possible portions of mechanisms for a resource group monopolizing task switching process are provided as functions of hardware to perform the process mainly by hardware processing. However, since a resource group monopolizing task switching process is not so frequently performed as a resource group usability check, it may be performed by software processing based on an operating system or the like. A large portion of a resource group monopolizing task switching process can be realized by using software such as an operating system, hardware provided in a data processing apparatus, or microprograms.

[0023] More specifically, in a resource group monopolizing task switching process, a determination is made as to whether it is necessary to save resource values of switching-object resource groups, and, if so, processing for saving th resource values and processing for updating flags necessary for determination as to whether saving of the resource values is necessary are performed. A determination is also made as to whether there is a need to read the resource values of the switching-object resource groups. If reading of the resourc values is required, processing for reading the resource values and processing for updating flags for determination as to whether there is a need to read the resource values are performed. Further, processing for updating variables necessary for identifying a task monopolizing some of the resource groups and processing for updating flags used in a resource group usability check, and so on are performed.

[0024] To perform such a resource group monopolizing task switching process, a resource group in-use flag corresponding to each of resource groups is prepared as a means for determination as to whether it is necessary to save resource values of switching-object resource groups. If during execution of a task a resource which belongs to one of the resource groups is used, the value "TRUE" is set in the corresponding resource group in-use flag. If, in a task executed by task switching, a processing with which destruction of resource values of a particular one of the resource groups is presupposed (e.g., a system call, a subroutine call or processing for return from a subroutine) is executed, the value "FALSE" is set in the resource group in-use flag corresponding to the resource group. Also, if it is determined that resource values of a resource used by a task can be discarded with no problem, for example, when execution of the task is stopped, the value "FALSE" is set in the resource group in-use flag corresponding to the resource group which has been monopolized by the task.

[0025] To enable a resource group monopolizing task switching process such as described above, resource group saved value validity flags corresponding to resource groups are prepared with respect to each of a plurality of tasks as means for making a determination as to whether there is a need to read resource values of switching-object resource groups. When an active task uses one of the resources, the value "TRUE" is set in the resource group saved value validity flag corresponding to the resource group to which the resource used by the task belongs, in the resource group saved value validity flags corresponding to the task. If the resource group saved value validity flags are used in this manner, a register for holding the resource group saved value validity flags corresponding to the active task may be provided in a data processing apparatus. Memory accessing events required for updating the resource group saved value validity flags can be reduced thereby.

[0026] If a task started by task switching to be executed includes a process in which destruction of resource values of a particular one of the resource groups is presupposed, the value "FALSE" is set in the resource gr up saved value validity flag corresponding to the resource group in which destruction of resource

values is allowed, in the resource group saved value validity flags corresponding to the task. In this manner, saving and reading of unnecessary resource values can be omitted to reduc memory accessing events. If it is determined that resource values of a resource used by a task can be discarded with no problem, for example, when execution of the task is stopped, the value "FALSE" is set in the resource group saved value validity flag corresponding to the resource group which has been monopolized by the task, in the resource group saved value validity flags corresponding to the task. In this manner, saving and reading of unnecessary resource values can be omitted to further reduce memory accessing events.

[0027] To perform a resource group monopolizing task switching process such as described above, resource group monopolizing task variables corresponding to resource groups are prepared as means for identifying a task monopolizing some of the resources. In the resource group monopolizing task switching process, the value of the resource group monopolizing task variable corresponding to each resource group is updated to the value corresponding to the active task.

2. Basic Processing Flow

[0028] When a need for a task switch has arisen, an operating system updates the resource group saved value validity flags and the resource group monopolization flags, etc., corresponding to a task to be stopped from being executed. The operating system also performs saving and setting of the resource group use permission flags, the resource group saved value validity flags, the active task variable, etc., according to need with respect to a task started by task switching to be executed. Further, the operating system performs saving and reading of resource values at suitable times in the process of saving and setting these flags, if necessary. After these processings, the operating system starts execution of the new task.

[0029] If the active task started by task switching is about to use a resource which belongs to a resource group which the task has not been permitted to use. The state of the task being about to use the resource in the resource group without permission is detected by a resource group usability check, and a resource group monopolizing switching process is then started. In the resource group monopolizing switching process, processings necessary for enabling the task to use the resource in the resource group that the task has not been permitted to use are performed. After the resource group monopolizing switching process has enabled the task to use the resource, the process of the task using the resource is continued.

[0030] If a register holding the resource group in-use flags has been provided in the form of hardware, th value "TRUE" is set in the resource group in-us flag corresponding to the resource group to which the resource required by the active task belongs when the active task is about to use the resource. If a register holding the resource group saved value validity flags has been provided in the form of hardware, the value "TRUE" is set in the resourc group saved value validity flag corresponding to the resource group to which the resource required by the active task belongs when the active task is about to use the resource.

[0031] In the resource group monopolizing task switching process, a task monopolizing the resource group to which the resource that the active task is about to use belongs is first identified from the resource group monopolizing task variable corresponding to the resource group. Then the value "FALSE" is set in the resource group monopolization flag corresponding to the task monopolizing the resource group. Also, the resource group saved value validity flag corresponding to the task monopolizing the resource group is checked. If the value of the resource group saved value validity flag is "TRUE", resource values of the resource group are saved in the resource value save area corresponding to the task monopolizing the resource group.

[0032] If a register holding the values of the resource group in-use flags has been provided in the form of hardware, the corresponding content of this register is checked. If the value of the resource group in-use flag corresponding to the switching-object resource group is "TRUE", resource values of the resource group are saved in the resource group save area corresponding to the task monopolizing the resource group.

[0033] Next, the resource group saved value validity flags corresponding to the active task are checked. If the value corresponding to the monopolizing task switching-object resource group is "TRUE", resource values of the resource group are read from the resource value save area corresponding to the active task to the resource, thereby enabling the active task to see the resource values as if the values that it set by itself have existed continuously in the resource, thus ensuring independence between the tasks.

[0034] If the value of the resource group saved value validity flag corresponding to the active task is "FALSE", a check is performed as to whether the resource value save area corresponding to the monopolizing task switching-object resource group has been secured. If no, the resource value save area corresponding to the resource group is secured. Thereafter, the value "TRUE" is set in the resource group saved value validity flag corresponding to the resource group.

[0035] Following the above-described process, the resource group monopolizing task variable corresponding to the monopolizing task switching-object resource group is set to the value corresponding to the active task. The process of switching the task monopolizing the resource group is thereby completed, thereby enabling the active task to use th resource.

[0036] In the case of performing resourc group usability check by using the resource group user permis-

sion flags, the resource group monopolizing task switching process also includes a process in which the value "TRUE" is set in the resource group use permission flag corresponding to the monopolizing task switching-object resource group.

3. About use of Memory

[0037] When task switching is performed in the above-described manner, it is preferable to secure the resource value save area for a task in one memory block in which a stack used by the task is also provided. Before a task is generated, it is not necessary to secure the memory block area used for its stack. Also, this memory block area can be freed when execution of the task is finished. Because of such a characteristic of a stack, it is possible to realize a mechanism for saving the memory consumption in such a manner that the memory block area for a stack is dynamically allocated for a task. If such a mechanism is used, and if the resource values save area and a stack used by the same task are secured in one common memory block, the resource value save area can also be dynamically allocated, thereby further saving the memory consumption. One resource value save area is provided for one task. When each resource value save area is secured, it is preferable to dynamically change the size of the resource value save area according to the size of resource values to be saved therein. In this manner, the memory can be used efficiently. In such memory allocation, while the size of the resource value save area is dynamically changed as described above, the size of the resource value save area is increased to accommodate resource values of a resource group newly used by an active task in the resource group monopolizing task switching process. In this event, it is preferred that the resource value save area is secured in a direction opposite from the direction in which data is stored in a stack by the task, thereby enabling further effective use of the memory.

[0038] When the resource value save area and a stack used by a task are secured in one common memory block, it is desirable to prevent overlapping of the resource value save area and the area used for a stack. To avoid overlapping of the resource value save area and the area used for a stack, memory allocation may be such that, when the size of the resource value save area is dynamically changed, occurrence of overlapping of the resource value save area with the area used as a stack is checked, and that, when the size of the area used as a stack by a task is dynamically changed, occurrence of overlapping of the area used as a stack with the resource value save area is checked.

[0039] If the data processing apparatus of at least preferred embodiments of the present invention has a cache memory, it is preferable t separate resources into resource groups in such a manner that the size of the resource value save area corresponding to each resource group is an integer multiple of the cache line size of the cache memory. If the size of the resource value save area corresponding to each resource group is an integer multiple of the cache line size of the cach memory, accessing to the memory can be performed more efficiently.

[0040] If the hardware includes a cache memory, and if resource value save areas are secured on the cache memory, it is preferred that the starting address of each resource value save area is set at a cache line boundary of the cache memory. If the starting address of each resource value save area corresponds to a cache line boundary of the cache memory, accessing to the memory can be performed more efficiently.

4. Embodiments of the Invention

[0041] Embodiments of the present invention will now be described in detail. An example of the task switching method of at least preferred embodiments of the present invention realized by a process based on an operating system with an existing processor will first be described as a first embodiment of the present invention, and an example of task switching using a novel data processor to which the present invention has been applied will thereafter be described as a second embodiment of the present invention.

4-1 First Embodiment

[0042] A case in which the task switching method of the present invention has been realized mainly by a process based on an operating system with conventional practical hardware will first be described as a first embodiment of the present invention.

[0043] The task switching method described below is mainly expressed as a process based on an operating system, which may have any configuration excepting a module for realizing task switching and, therefore, may have the same configuration as the conventional operating systems.

[0044] For ease of explanation in the following description of the task switching method, it is assumed here that the operating system for executing a task switching process and so on performs processing by using registers for the operating system only and does not destroy values in registers used by tasks. However, the present invention can also be applied to a process in which an operating system and tasks use the same registers. In a case where an operating system and tasks use the same registers, saving and reading of values in the registers may be performed by suitable timing according to whether each register is used by the operating system or the task.

4-1-1 Hardwar Configuration

[0045] Hardware used in this embodiment will first be

described. The hardware used in this embodiment is not a novel one but one already put to practical use.

[0046] The hardware used in this mbodiment includes a central processing unit (CPU) and a floating-point arithmetic unit for executing instructions of operations such as floating-point operations at a high speed. The central processing unit has a cache memory having a cache line size of 16 bytes.

[0047] As shown in Fig. 1, this hardware has, as resources usable for tasks, registers for the central processing unit (hereinafter referred to as general registers) and registers for the floating-point arithmetic unit (hereinafter referred to as floating-point registers).

[0048] The registers for the central processing units are general registers 0 to 127. Each general register has a capacity of 32 bits. The registers for the floating-point arithmetic unit are floating-point registers 0 to 63. Each floating-point register has a capacity of 64 bits. This hardware has a total capacity of 8192 bits (= 1024 bytes), which is the sum of capacities of the general registers and floating-point registers.

[0049] This hardware also has a 1-bit register in which is stored a use permission flag for indication as to whether the floating-point arithmetic unit is usable or unusable (hereinafter referred to as a floating-point arithmetic unit use permission flag).

4-1-2 Program Specifications

[0050] In this embodiment, a program executed by the hardware is specified as described below. When a functional block call or return instruction, such as a subroutine call, an instruction for a return from a subroutine or a system call (these sorts of instructions hereinafter referred to collectively as "functional block transition instruction" for simplification), is executed, the general registers 0 to 63 and the floating-point registers 0 to 31 are used as work registers, and the values thereof can be changed during processing with a called functional block. On the other hand, it is necessary that, when the process of the program is advanced by finishing the processing with the functional block, the general registers 64 to 127 and the floating-point registers 32 to 63 be holding the values held before the call of the functional block.

[0051] A demand for a task switch from an active task to the operating system is made by using a functional block transition instruction.

[0052] To enable the process to allow interrupts generated at arbitrary times, it is necessary that, following interrupt processing, the main process be restarted after restoring the values of all the general registers and the floating-point registers to the values held immediately before the interrupt.

4-1-3 Outline of Operating System and Data Structure

[0053] An outline of the operating system and the structure of data handled by the operating system will next be described.

[0054] The operating system used in this embodiment has a multitasking function. That is, in this operating system, resource values representing the statuses of resources used by tasks (i.e., the values of the general registers and the floating-point registers) are saved in resource value save areas respectively prepared for the tasks, and the resource values of the resources used by one of the tasks when the task is executed are read out from the resource value save area, thereby enabling a plurality of tasks to be executed in parallel with each other while ensuring independence of each task.

[0055] At least preferred embodiments of the present invention are characterized in that, to realize such a multitasking function, resources usable by tasks are controlled by being separated into a plurality of resource groups. In this embodiment, the general registers and the floating-point registers, which are resources usable by tasks through the operating system, are controlled by being separated into four resource groups: a resource group 0, a resource group 1, a resource group 2, and a resource group 3, as shown in Fig. 2.

[0056] The resource group 0 is a group of general registers with no need to hold, at the time of task switching by a functional block transition instruction, the values held before the functional block transition instruction is issued. That is, the general registers 0 to 63 belong to the resource group 0. The resource group 1 is a group of general registers with a need to hold, at the time of task switching by a functional block transition instruction, the values held before the functional block transition instruction is issued. That is, the general registers 64 to 127 belong to the resource group 0. The resource group 2 is a group of floating-point registers with no need to hold, at the time of task switching by a functional block transition instruction, the values held before the functional block transition instruction is issued. That is, the floating-point registers 0 to 31 belong to the resource group 2. The resource group 3 is a group of floating-point registers with a need to hold, at the time of task switching by a functional block transition instruction, the values held before the functional block transition instruction is issued. That is, the floating-point registers 32 to 63 belong to the resource group 3.

[0057] In the operating system, a resource groups 2-3 monopolizing task variable and an active task variable are defined as variables in a memory, as shown in Fig. 2. The resource groups 2-3 monopolizing task variable is set to a value designating a task which is monopolizing the resource groups 2 and 3. That it, the resource values of the resource groups 2 and 3 are valid with respect to a task designated by the resource group 2-3 monopolizing task variable. The active task variable is set to a value designating a task which is presently being executed. That is, in this embodiment, a task which is being executed can be id ntified by referring to the value of the active task variable.

[0058] In the operating system, a floating-point arithmetic unit use permission flag provided in the hardware is used as a resourc group use permission flag for indication as to whether use of the resources which belong to the resource groups 2 and 3 is permitted. More specifically, if an active task can use the resources which belong to the resource groups 2 and 3, the floating-point arithmetic unit use permission flag is set to the value "TRUE". If an active task is not to be permitted to use the resources which belong to the resource groups 2 and 3, the floating-point arithmetic unit use permission flag is set to the value "FALSE".

[0059] As shown in Fig. 3, to execute a plurality of tasks in parallel with each other, the operating system maintains, in a memory, for each task, a task information save area for saving task information and a resource value save area for saving resource values indicating the status of some of the resources used in execution of the task.

[0060] One set of information items corresponding to one task is stored as task information in each task information save area. Sets of task information items corresponding to tasks are collectively saved in array in the memory according to numbering of the tasks, as shown in Fig. 4. Each information set can be referred to by using the corresponding suffix in the array. That is, each task information set is saved in the memory as an array element, and the area in which each array element is saved corresponds to the task information save area. If task information sets are saved in array in the memory in this manner, the memory can be effectively used and each task information set can be accessed in a short time.

[0061] Each task information save area includes a task control information area and a resource control information area, as shown in Fig. 5. The task control information area is an area in which task control information required by the operating system to control a task is saved. Task control information saved in the task control information area is not a feature of the present invention. It may be the same as that used in the conventional systems and will not be described in detail.

[0062] The resource control information area is an area in which resource control information necessary for controlling some of the resources used by a task is saved, and which has not been provided in the conventional systems. In the resource control information area are saved a resource groups 2-3 monopolization flag, resource group saved value validity flags, and a resource value save area starting address.

[0063] The resource groups 2-3 monopolization flag in these resource control information items is a flag for indication as to whether the resource groups 2 and 3 are monopolized. When a task is monopolizing the resource groups 2 and 3, the resource groups 2-3 monopolization flag in the task information corresponding to the task is set to the valu "TRUE". When the task is not monopolizing the resource groups 2 and 3, the resource groups 2-3 monopolization flag in the task information corresponding to the task is set to the value "FALSE".

[0064] That is, if the resource groups 2-3 monopolization flag in the task information corresponding to a task is "TRUE", the resource values of the resource groups 2 and 3 are valid with respect to the task. If the resource groups 2-3 monopolization flag in the task information corresponding to a task is "FALSE", the resource values of the resource groups 2 and 3 are invalid with respect to the task.

[0065] The resource group saved value validity flags in the resource control information items are flags for indication as to whether some valid values have been saved in the resource value save area. The resource group saved value validity flags are provided in one-to-one correspondence with the resource groups.

[0066] If valid values have been saved in the resource value save area as resource values of one of the resource groups, the corresponding resource group saved value validity flag is set to the value "TRUE". If no valid values have been saved, the flag is set to the value "FALSE". For example, if valid values have been saved as resource values of the resource group 2, the resource group saved value validity flag corresponding to the resource group 2 is set to the value "TRUE". If no valid values have been saved as resource values of the resource group 2, the resource group saved value validity flag corresponding to the resource group 2 is set to the value "FALSE".

[0067] When execution of a task is completed, saving of resource values in the resource value save area is not performed and the values of all the resource group saved value validity flags in the task information corresponding to the task are set to the value "FALSE", thereby avoiding saving of unnecessary resource values.

[0068] A value designating the starting address of each resource value save area in the memory is set as the above-mentioned resource value save area starting address in resource control information. Conventionally, a task information save area and a resource value save area are provided as a set of adjacent areas in a memory, as shown in Fig. 20. In contrast, in this embodiment of the present invention, each task information save area and the corresponding resource value save area are maintained separately from each other in the memory, as shown in Figs. 3 and 5. In this embodiment, therefore, the resource value save area starting address is saved as resource control information for access to each resource value save area.

[0069] As mentioned above, the operating system maintains a resource value save area for each task. In each resource value save area, the resource values indicating the status of some of the resources used for execution of th task corresponding to the resource value save area ar saved. In this embodiment, however, the resource values of all th resources are not

always saved in each resource value save area, and the resource values of necessary ones of the resources, collectively selected as some of the resource groups, are saved in each resource value save area.

[0070] That is, if it is sufficient to save, for example, only the resource values of the resource groups 0 and 1 for a task, only the resource values of the resource groups 0 and 1 are saved in the resource value save area corresponding to the task, as shown in Fig. 5. In other words, with respect to a task not using, for example, the floating-point arithmetic unit, there is no need for resource value save area for the resource groups 2 and 3.

[0071] In the conventional task switching method, the resource values of all resources are saved in the resource value save area for each of executable tasks, so that the amount of memory for all the general registers and floating-point registers, i.e., an amount of 1024 bytes, has been required as the resource value save area for one task. On the other hand, in this embodiment, resources are controlled by being separated into resource groups, and only the resource values of the resource groups to which some of the resources the values of which are to be saved belongs are stored in each resource value save area. With respect to tasks not using, for example, the floating-point arithmetic unit, there is no need for the resource value save area corresponding to the floating-point registers. Consequently, an amount of memory of 512 bytes (i.e., the amount of memory corresponding to the floating-point registers) per task can be saved in comparison with the conventional task switching method.

[0072] In this embodiment, as described above, resources are controlled by being separated into resource groups, and only the resource values of the resource groups to which some of the resources the values of which are to be saved belongs are stored in each resource value save area. Therefore, the amount of data to be saved in the memory when a task switch is made can be effectively reduced, and the number of memory accessing events or the amount of data transferred by accessing the memory at the time of task switching can be effectively reduced.

[0073] When the resource values of some of the resources are stored in one resource value save area, the resource values of the resource group 0 are successively stored from the memory address designated by the resource value save area starting address in the resource control information. The resource values of the resource group 1 are successively stored from the memory address 256 bytes behind the memory address designated by the resource value save area starting address in the resource control information. The resource values of the resource group 2 are successively stored from the memory address 512 bytes behind the memory address designated by the resource value save area starting address in the resource control information. Finally, the resource values of the resource group 3 are successively stored from the memory address 768 bytes behind the memory address designated by the resource value sav area starting address in th resource control information.

[0074] If the relationship between the relative positions of subareas for saving the resource values of the resource groups is predetermined in this manner, the resource value save area starting address may suffice as information indicating the subarea for saving the resource values of each resource group, and the need for storing the memory address of the subarea for saving the resource values of each resource group can be eliminated.

4-1-4 Outline of Task Switching Process

[0075] A task switching process performed by the above-described operating system will next be outlined.

[0076] In task switching of this embodiment, with respect to resource groups 2 and 3 controlled with the floating-point arithmetic unit use permission flag, saving of the resource values in the relating resource value save area and reading of the resource values from the relating resource value save area are not performed at the time of task switching. With respect to resource groups 2 and 3, saving and reading of the resource values are performed if necessary when a need for some of the resources which belong to resource group 2 or 3 arises.

[0077] Also with respect to resource groups 0 and 1, saving of the resource values in the relating resource value save area and reading of resource values from the relating resource value save area are not always performed. Saving and reading of the resource values may be performed only with respect to a necessary one of resource groups 0 and 1.

[0078] That is, if task switching is started by a functional block transition instruction, it is necessary to hold the resource values of resource group 1 but it is possible to change the resource values of resource group 0. In this case, therefore, only the resource values of resource group 1 are saved. On the other hand, if task switching is started by interrupt processing, it is necessary to hold the resource values of both the resource groups 0 and 1, and the resource values of both resource groups 0 and 1 are saved in the resource value save area.

[0079] To read resource values from each resource value save area, the resource group saved value validity flag is first checked to ascertain whether values have been stored in the resource value save area as valid resource values of resource group 0 or 1. Only if valid values are stored, reading of resource values from the resource value save area is performed. That is, with respect to a task to be restarted by task switching, only valid resource values ar read from the resourc value save area by referring to the value of the resource group saved value validity flag set when execution of the task

BNSDOCID: <EP 0953905A2 I >

BNS page 1(

was stopped.

[0080] As mentioned above, with respect to resource groups 2 and 3, saving of resource values in the resource value save area and reading of resource values from the resource value save area are not performed at the time of task switching. If a need for some of the resources which belong to resource group 2 or 3 has arisen, a resource group usability check is first executed. In the resource group usability check, an examination is made as to whether the task can use the resource which belongs to resource group 2 or 3, as described below in detail.

[0081] If the result of the resource group usability check is that the task can use the resource, the process of the task is continued by using the resource. If it is determined that the task cannot use the resource, a resource group monopolizing task switching process is performed. In the resource group monopolizing task switching process, the state of the system is changed so that the task can use the resource, described below in detail. After the state of the system has been changed by the resource group monopolizing task switching process to enable the task to use the resource, the process of the task is continued by using the resource.

4-1-5 Details of Task Switching Process

[0082] The task switching process will be described in detail with reference to the flowchart of Fig. 6. The description will be made with respect to a task switch from task A to task B.

[0083] In the process of task switching from task A to task B, it is first determined in step ST1-1 as to whether task switching is started by interrupt processing. If task switching is started by interrupt processing, the process advances to step ST1-2. If task switching is not started by interrupt processing (that is, if task switching is started by a functional block transition instruction), the process advances to step ST1-4.

[0084] In step ST1-2, setting of resource group saved value validity flags in the task information corresponding to task A is performed for task switching started by interrupt processing. More specifically, the values of the resource group saved value validity flags corresponding to resource groups 0 and 1 in the resource group saved value validity flags in the task information corresponding to task A are set to "TRUE". The values of the resource group saved value validity flags corresponding to resource groups 2 and 3 in the resource group saved value validity flags in the task information corresponding to task A are not changed.

[0085] Next, in step ST1-3, the resource values of the resource groups without the use permission flag are saved in the resource value save area corresponding to task A. More specifically, the resource values of resource groups 0 and 1 are saved in the resourc valu save area corresponding to task A. With respect to resource groups 2 and 3, a resource group usability check can be executed by using the floating-point arithmetic unit use permission flag, as described below. At the time of task switching, therefor , saving of the resource values of resource groups 2 and 3 is not performed. Then, after completion of the processing in step ST1-3, the process advances to step ST1-6.

[0086] On the other hand, in step ST1-4, setting of resource group saved value validity flags in the task information corresponding to task A is performed for task switching performed by a functional block transition instruction. More specifically, the value of the resource group saved value validity flag corresponding to resource group 0 is set to "FALSE", the resource group saved value validity flag corresponding to resource group 1 is set to "TRUE", and the value of the resource group saved value validity flag corresponding to resource group 2 is set to "FALSE". The value of the resource group saved value validity flag corresponding to resource group 3 is not changed.

[0087] In step ST1-5, the resource values of the resource groups without the use permission flag are saved to the resource value save area corresponding to task A. More specifically, the resource values of resource group 1 are saved in the resource value save area corresponding to task A. Although the resource groups without the use permission flag are resource groups 0 and 1, the resource values of resource group 0 are not saved in this embodiment because the resource values of resource group 0 can be changed in the case of task switching started by a functional block transition instruction. Since a resource group usability check can be executed by using the floating-point arithmetic unit use permission flag with respect to resource groups 2 and 3 as described below, saving of the resource values of resource groups 2 and 3 is not performed at the time of task switching. After completion of the processing in step ST1-5, the process advances to step ST1-6.

[0088] Step ST1-6 and other subsequent steps are common task switching started by interrupt processing and task switching started by a functional block transition instruction. First, in step ST1-6, the value of floating-point arithmetic unit use permission flag is set as the resource group 2-3 monopolization flag in the task information corresponding to task A (i.e., switched-off task).

[0089] If the resource values of resource groups 0 and 1 have been saved in the resource value save area corresponding to task B (i.e., switched-on task), they are read out by processings in steps step ST1-7 to step ST1-10. That is, while there is no need for reading the resource values of resource groups 2 and 3 at the time of task switching since a resource group usability check can be executed with respect to resource groups 2 and 3, reading of the resource values of resource groups 0 and 1 is performed at the time of task switching by referring to the resourc group saved value validity flag corresponding to task B since resource gr up usability checking cannot be p rformed with respect to resource

BNSDOCID: <EP 0953905A2 1 > BNS page 1

groups 0 and 1.

[0090] In more detail, in step ST1-7, the value of resource group 0 saved value validity flag in the resourc group saved value validity flag in the information corresponding to task B is checked. If the value of the resource group 0 saved value validity flag is "TRUE", the process advances to step ST1-8 and the resource values of resource group 0 are read. The process then advances to step ST1-9. If the value of the resource group 0 saved value validity flag is "FALSE", the process moves to step ST1-9 without reading of the resource values of resource group 0.

[0091] In step ST1-9, the value of resource group 1 saved value validity flag in the resource group saved value validity flag in the information corresponding to task B is checked. If the value of the resource group 1 saved value validity flag is "TRUE", the process advances to step ST1-10 and the resource values of resource group 1 are read. The process then advances to step ST1-11. If the value of the resource group 1 saved value validity flag is "FALSE", the process moves to step ST1-11 without reading of the resource values of resource group 0.

[0092] In step ST1-11, the value of the resource group 2-3 monopolization flag in the task information corresponding to task B is set as the floating-point arithmetic unit use permission flag.

[0093] In step ST1-12, the active task variable is set to value corresponding to task B. The processing necessary for control of the resource groups and the processing for necessary saving and reading of the resource values at the time of task switching are completed by the above-described steps.

[0094] Finally, in step ST1-13, other task switch processings are performed. After all the task switch processings have been completed, task B is started. The processings in step ST1-13 are processings other than those for resource management and may be the same as those of the conventional operating systems. Therefore, they do not relate particularly to the essentials of the present invention and will not be described specially in detail.

[0095] Processing similar to that in step ST1-12 is usually performed to control executed tasks in task switching processes with the conventional operating systems. However, the active task variable is a variable used in the process specific to the present invention, and the processing for setting the active task variable to the value corresponding to task B is therefore shown as a step independent of step ST1-13.

[0096] According to the conventional task switching method, if task switching is performed in a data processor having a configuration such as shown in Fig. 1, there is a need for saving and reading of values with respect to all general registers and floating-point registers, i.e., saving of resource values of 1024 bytes in a resource value save area and reading of resource values of 1024 bytes from the resource value save area. In contrast, in the above-described task switching process, resources are controlled by being separated into four resource groups 0, 1, 2, and 3, and saving to the resource value save area and reading from the save area are performed with respect to only necessary ones of the resource groups, so that the number of memory accessing events or the amount of data transferred by accessing the memory at the time of task switching is markedly small.

[0097] That is, in the above-described task switching process, if task switching is started by interrupt processing, saving of resource value in the resource value save area corresponding to a switched-off task (i.e., task A) may be performed with respect to resource groups 0 and 1 only. In this case, only a space of 512 bytes suffices as the memory area for saving resource values. If task switching is started by a functional block transition instruction, saving of resource values in the resource value save area corresponding to a switched-off task (i.e., task A) may be performed with respect to resource group 1. In this case, only a space of 256 bytes suffices as the memory area for saving resource values.

[0098] In the above-described task switching process, the maximum amount of resource value data read from the resource value save area corresponds to the case where each of the resource group 0 saved value validity flag and the resource group 1 saved value validity flag in the resource group saved value validity flags in the task information corresponding to the switched-on task (i.e., task B) is "TRUE". Even in this case, reading of resource value from the resource value save area corresponding to the switched-on task may be performed with respect to resource groups 0 and 1 only. That is, the amount of resource value data read from the resource value save area in the above-described task switching process is only 512 bytes at the maximum.

[0099] If, in the above-described task switching process, the resource group 0 saved value validity flag and the resource group 1 saved value validity flag in the resource group saved value validity flags in the task information corresponding to the switched-on task (i.e., task B) are "FALSE" and "TRUE", respectively, reading of resource values from the resource value save area corresponding to the switched-on task may be performed with respect to resource group 1 only. In this case, the amount of resource value data read from the resource value save area is only 256 bytes.

[0100] The case where each of the resource group 0 saved value validity flag and the resource group 1 saved value validity flag in the resource group saved value validity flags in the task information corresponding to the switched-on task (i.e., task B) is "TRUE" is a case where the switched-on task has been stopped by task switching started by interrupt processing. The case where the resource group 0 saved value validity flag and the resource group 1 saved value validity flag in the resource group saved value validity flags in the task information corresponding to the switched-on task (i.e.,

BNSDOCID: <EP 0953905A2 I >

task B) are "FALSE" and "TRUE", respectively, is a case where th switched-on task has been stopped by task switching started by a functional block transition instruction.

4-1-6 Resource Group Usability Check

[0101] Resource group usability check will be described in detail with reference to the flowchart of Fig. 7.

[0102] A resource group usability check is performed in parallel with execution of an instruction with the central processing unit. That is, a resource group usability check is executed each time an instruction for a task is executed.

[0103] First, in step ST2-1 of the process for a resource group usability check, a determination is made as to whether an instruction to be executed requires use of the floating-point arithmetic unit. If the instruction requires use of the floating-point arithmetic unit, the process advances to step ST2-2. If the instruction does not require use of the floating-point arithmetic unit, the resource group usability check is immediately finished and the instruction for the task is executed without using the floating-point arithmetic unit.

[0104] In step ST2-2, the value of the floating-point arithmetic unit use permission flag is checked. If the value of the floating-point arithmetic unit use permission flag is "FALSE", the process advances to step ST2-3. If the value of the floating-point arithmetic unit use permission flag is "TRUE", the conditions for use of the floating-point arithmetic unit by the task have already been satisfied. The resource group usability check is then finished and the instruction for the task is executed by using the floating-point arithmetic unit.

[0105] Since the process advances to step ST2-3 when the conditions for use of the floating-point arithmetic unit by the task are not satisfied, the resource group monopolizing task switching process is started in step ST2-3 to enable the task to use the floating-point arithmetic unit, as described below in detail. When the resource group monopolizing task switching process is completed to enable the task to use the floating-point arithmetic unit, the resource group usability check is finished and the instruction for the task using the floating-point arithmetic unit is executed. Specifically, the process may be such that, if, when an instruction involving use of the floating-point arithmetic unit is executed (ST2-1), the floating-point arithmetic unit use permission flag is "FALSE" (ST2-2), an interrupt is caused and the process advances to resource group monopolizing task switching process (ST2-3).

[0106] The above-described resource group usability check is performed in parallel with execution of an instruction with the central processing unit, and the processing time is not increased by the resource group usability check if the resource group monopolizing task switching process is not started. That is, with respect to a task not using the floating-point arithmetic unit, the resource group monopolizing task switching process is not started and there is no influence of the resourc group usability check upon the task execution speed. Tasks not using the floating-point arithmetic unit, therefore, can be executed with the effect of reducing the processing time and the resource value save area by avoiding saving and reading resource values relating to the floating-point arithmetic unit, without making any sacrifice.

[0107] On the other hand, with a task using the floating-point arithmetic unit, there is a possibility of the resource group monopolizing task switching process being started. If the resource group monopolizing task switching process is started, the total processing load is increased by an amount corresponding to this process in comparison with the conventional art.

[0108] In this embodiment, however, saving in the resource value save area and reading from the resource value save area of the floating-point register values (i.e., the resource values of the resource groups 2 and 3) are not performed at the time of switching to a task not using the floating-point arithmetic unit. The total processing load is reduced by the corresponding amount in comparison with the conventional art. For example, a situation may be taken into consideration in which, during the time period from a moment at which a task using the floating-point arithmetic unit is stopped to a moment at which execution of this task is restarted, no other task uses the floating-point arithmetic unit, and the values of the floating-point arithmetic unit registers are maintained during the time period according to at least preferred embodiments of the present invention. Therefore, processings required for or accompanying updating of the floating-point register values can be effectively reduced in comparison with the conventional task switching method in which all resource values are updated each time task switching is performed.

[0109] As long as the frequency of task switching between tasks each using floating-point arithmetic unit is not excessively high, the effect of reducing the processing load by eliminating processings required for or accompanying updating of the floating-point register values advantageously prevails over the effect of increasing the processing load with the resource group monopolizing task switching process. Consequently, even if tasks existing as switching objects include tasks using floating-point arithmetic unit, the processing load in this embodiment is much smaller than that in the conventional art as long as the frequency of task switching between the tasks using floating-point arithmetic unit is not excessively high. 4-1-7 Resource Group Monopolizing Task Switching Process

[0110] The resource group monopolizing task switching process will be described in detail with reference to the flowchart of Fig. 8.

[0111] The resource group monopolizing task switching process is executed in th case where, when a task

BNSDOCID: <EP 0953905A2 I >

is about to use a resource (the floating-point registers in this embodiment), a resource group usability check is executed to determine that the resource is unusabl , as mentioned above. That is, if a resource is unusabl when a task is about to use the resource, the relating conditions are changed by the resource group monopolizing task switching process so that the resource becomes usable by the task. The resource group monopolizing task switching process comprises two main processings, i.e., processing for changing a right to use of some resource group, and processing for saving and reading of the resource values of the resource group.

[0112] This resource group monopolizing task switching process is performed as shown in the flowchart of Fig. 8. First, in step ST3-1, the task information corresponding to the task designated by the resource groups 2-3 monopolizing task variable (i.e., the task information about the task presently monopolizing the floating-point arithmetic unit) is identified as monopolizing task information.

[0113] In step ST3-2, the resource groups 2-3 monopolization flag in the monopolizing task information is set to the value "FALSE".

[0114] In step ST3-3, the floating-point arithmetic unit use permission flag is set to the value "TRUE", thereby permitting a presently executed task to use the floating-point arithmetic unit.

[0115] If it is necessary to hold the resource values of resource groups 2 and 3, the resource values are saved in the resource value save area corresponding to the task presently monopolizing the resource groups 2 and 3 by processings in steps ST3-4 to ST3-7.

[0116] More specifically, in step ST3-4, the resource group 2 saved value validity flag in the resource group saved value validity flags of the monopolizing task information is checked. If the value of the resource group 2 saved value validity flag is "TRUE", the process advances to step ST3-5 and the resource values of resource group 2 are saved in the resource value save area corresponding to the monopolizing task information. The process then advances to step ST3-6. If the value of resource group 2 saved value validity flag is "FALSE", the process moves to step ST3-6 without saving the resource values of resource group 2 in the resource value save area.

[0117] In step ST3-6, the resource group 3 saved value validity flag in the resource group saved value validity flags of the monopolizing task information is checked. If the value of the resource group 3 saved value validity flag is "TRUE", the process advances to step ST3-7 and the resource values of resource group 3 are saved in the resource value save area corresponding to the monopolizing task information. The process then advances to step ST3-8. If the value of resource group 3 saved value validity flag is "FALSE", the process moves to step ST3-8 without saving the resource values of resource group 3 in the resource value save area.

[0118] In step ST3-8, the task information corresponding to the task designated by the active task variable (i.e., the task information corresponding to the task which is presently being executed) is identified as active task information.

[0119] Next, if valid values have been saved as resource group 2 and/or 3 resource values in the resource value save area corresponding to the active task, the saved values are read by processings in steps ST3-9 to ST3-12.

[0120] More specifically, in step ST3-9, the resource group 2 saved value validity flag in the resource group saved value validity flags of the active task information is checked. If the value of the resource group 2 saved value validity flag is "TRUE", the process advances to step ST3-10 and the resource values of resource group 2 are read from the resource value save area corresponding to the active task information. The process then advances to step ST3-11. If the value of resource group 2 saved value validity flag is "FALSE", the process moves to step ST3-11 without reading the resource values of resource group 2 from the resource value save area.

[0121] In step ST3-11, the resource group 3 saved value validity flag in the resource group saved value validity flags of the active task information is checked. If the value of the resource group 3 saved value validity flag is "TRUE", the process advances to step ST3-12 and the resource values of resource group 3 are read from the resource value save area corresponding to the active task information. The process then advances to step ST3-13. If the value of resource group 3 saved value validity flag is "FALSE", the process moves to step ST3-13 without reading the resource values of resource group 3 from the resource value save area.

[0122] In step ST3-13, each of the resource group 2 saved value validity flag and the resource group 3 saved value validity flag in the resource group saved value validity flags in the active task information is set to the value "TRUE". Even though the use of the resource groups 2 and 3 is permitted with respect to the active task, the active task does not necessarily use both the resource groups 2 and 3. Accordingly, even though the use of the resource groups 2 and 3 is permitted with respect to the active task, it does not necessarily follow that there is a need to hold both the resource values of the resource group 2 and the resource values of the resource group 3. In this embodiment, however, no means is provided for determination as to whether a task has actually used some of the resources which belong to the resource group 2 or 3. In this embodiment, it is assumed that there is a need to hold all the resource values of resources which have been rendered usable. Accordingly, in step ST3-13, each of the values of the resource group 2 save validity flag and the resource gr up 3 save validity flag is set to "TRUE".

[0123] Finally, in step ST3-14, the resource groups 2-

BNSDOCID: <EP 0953905A2 I >

3 monopolizing task variable is set to the valu of the active task variable.

[0124] The resource group monopolizing task switching process is thus completed, thereby enabling the active task to use the resourc groups 2 and 3 (i.e., the floating-point registers).

4-2 Second Embodiment

[0125] Task switching performed by using a novel data processing apparatus to which the present invention has been applied will now be described as a second embodiment of the present invention.

[0126] In this embodiment, registers for holding necessary information for task switching are provided in the form of hardware, and a mechanism for switching tasks by using the registers is also provided as functions of hardware corresponding to major portions of processings required for or accompanying task switching.

[0127] If such a task switching mechanism is provided as functions of hardware, the amount of software processing performed by the operating system can be effectively reduced and task switching can be efficiently performed at a higher speed in comparison with the case of performing task switching only by the operation of the operating system.

4-2-1 Hardware Configuration

[0128] Hardware used in this embodiment will first be described. The hardware of this embodiment is novel hardware to which the present invention has been applied.

[0129] The hardware used in this embodiment includes a central processing unit (CPU) and a floating-point arithmetic unit for executing instructions of operations such as floating-point operations at a high speed. The central processing unit has a cache memory having a cache line size of 16 bytes.

[0130] As shown in Fig. 9, this hardware has, as resources usable for tasks, registers for the central processing unit (hereinafter referred to as general registers) and registers for the floating-point arithmetic unit (hereinafter referred to as floating-point registers). The registers for the central processing units are general registers 0 to 127. Each general register has a capacity of 32 bits. The registers for the floating-point arithmetic unit are floating-point registers 0 to 63. Each floating-point register has a capacity of 64 bits. This hardware has a total capacity of 8192 bits (= 1024 bytes), which is the sum of capacities of the general registers and floating-point registers.

[0131] In this embodiment, a program executed by the hardware is specified as described below, as is that in the above-described first embodiment of the present invention. When a functional block transition instruction is executed, the general registers 0 to 63 and the floating-point registers 0 to 31 are used as work regist rs, and the values thereof can be changed during processing with a called functional block. On the other hand, it is necessary that, when the process of th program is advanced by finishing the processing with the functional block, the general registers 64 to 127 and the floating-point registers 32 to 63 be holding the values held before the call of the functional block.

[0132] A demand for a task switch from an active task to the operating system is made by using a functional block transition instruction.

[0133] To enable the process to allow interrupts generated at arbitrary times, it is necessary that, following interrupt processing, the main process be restarted after restoring the values of all the general registers and the floating-point registers to the values held immediately before the interrupt.

[0134] At least preferred embodiments of the present invention are characterized in that the general registers and the floating-point registers, which are resources usable by tasks, are controlled by being separated into sixteen resource groups, as shown in Fig. 9.

[0135] That is, this task switching apparatus controls the general registers and the floating-point registers, which are resources usable by tasks, by separating the tasks into the following sixteen resource groups.

Resource group 0: general registers 0 to 15
Resource group 1: general registers 16 to 31
Resource group 2: general registers 32 to 47
Resource group 3: general registers 48 to 63
Resource group 4: general registers 64 to 79
Resource group 5: general registers 80 to 95
Resource group 6: general registers 96 to 111
Resource group 7: general registers 112 to 127
Resource group 8: floating-point registers 0 to 7
Resource group 9: floating-point registers 8 to 15
Resource group 10: floating-point registers 16 to 23
Resource group 11: floating-point registers 24 to 31
Resource group 12: floating-point registers 32 to 39
Resource group 13: floating-point registers 40 to 47
Resource group 14: floating-point registers 48 to 55
Resource group 15: floating-point registers 56 to 63

[0136] The resource groups 0, 1, 2, and 3 are groups of general registers with no need to hold, at the time of task switching by a functional block transition instruction, the values held before the functional block transition instruction is issued. The resource groups 4, 5, 6, and 7 are groups of general registers with a need to hold, at the time of task switching by a functional block transition instruction, the values held before the functional block transition instruction is issued. The resource groups 8, 9, 10, and 11 are groups of floating-point registers with no need to hold, at the time of task switching by a functional block transition instruction, the values held before the functional block transition instruction is issued. The resource groups 12, 13, 14, and 15 are groups of floating-point registers with a need to hold, at

BNSDOCID: <EP 0953905A2 I >

the time of task switching by a functional block transition instruction, the values held before the functional block transition instruction is issued.

[0137] As shown in Fig. 9, the hardware of this task switching apparatus comprises registers for holding information necessary for task switching: resource group monopolizing task variable registers, an active task variable register, a resource group use permission flag register, a resource group in-use flag register, and a resource group saved value validity flag register. These registers, without their equals in the conventional hardware, are uniquely provided in the hardware shown in Fig. 9 to which the present invention has been applied.

[0138] The resource group monopolizing task variable registers are 32-bit unsigned integer registers provided in one-to-one correspondence with the sixteen resource groups. A resource group monopolizing task variable is set in each of the resource group monopolizing task variable registers to designate, if there is a task monopolizing the corresponding resource group, the starting address of an area in which task information corresponding to the task is stored. For example, if there is a task monopolizing the resource group 10, the starting address of an area in which task information corresponding to the task is stored is set as resource group monopolizing task variable 10.

[0139] The active task variable register is a 32-bit unsigned integer register in which an active task variable is stored. The active task variable is a variable designating the starting address of an area in which task information corresponding to a task which is presently being executed is stored. That is, when a task is being executed, the starting address of an area in which task information corresponding to the executed task is stored is set as an active task variable in the active task variable register.

[0140] The resource group use permission flag register is a 16-bit unsigned integer register in which sixteen resource group use permission flags are stored. Sixteen bits constituting the resource group use permission flag register are respectively used as sixteen independent resource group use permission flags corresponding to the sixteen resource groups formed as described above. Each of the resource group use permission flags is used for indication as to whether use of the corresponding resource group is permitted.

[0141] In each resource group use permission flag, the value "TRUE" is set if the corresponding resource group is usable by an active task, and the value "FALSE" is set if the resource group is unusable. For example, if the resource group 2 is usable, the value "TRUE" is set in the resource group use permission flag corresponding to the resource group 2. If the resource group 5 is unusable, the value "FALSE" is set in the resource group use permission flag corresponding to the resource group 5.

[0142] The resource group in-use flag register is a 16-bit unsigned integer register in which sixteen resource group in-use flags are stored. Sixteen bits constituting the resource group in-use flag register are respectively used as sixteen independent resource group in-use flags corresponding to the sixteen resource groups. Each of the resource group in-us flags is used for indication as to whether there is a task presently using the corresponding resource group.

[0143] In each resource group in-use flag, the value "TRUE" is set if some task is using the corresponding resource group, and the value "FALSE" is set if no task is using the corresponding resource group. For example, if the resource group 4 is in use, the value "TRUE" is set in the resource group in-use flag corresponding to the resource group 4. If the resource group 10 is not in use, the value "FALSE" is set in the resource group in-use flag corresponding to the resource group 10.

[0144] The resource group saved value validity flag register is a 16-bit unsigned integer register in which sixteen resource group saved value validity flags are stored. Sixteen bits constituting the resource group saved value validity flag register are respectively used as sixteen independent resource group saved value validity flags corresponding to the sixteen resource groups. Each of the resource group saved value validity flags is used for indication as to whether, with respect to an active task, effective resource values of the corresponding resource group exist in the resources of the resource group or in the corresponding resource value save area.

[0145] In each resource group saved value validity flag, the value "TRUE" is set if, with respect to an active task, effective resource values of the corresponding resource group have been held by the resources of the resource group or saved in the resource value save area, and the value "FALSE" is set if no valid values have been saved. For example, if valid resource values of the resource group 6 have been saved, the value "TRUE" is set in the resource group saved value validity flag corresponding to the resource group 6. If no valid resource values of the resource group 15 have been saved, the value "FALSE" is set in the resource group saved value validity flag corresponding to the resource group 15.

[0146] The hardware is arranged so that the operating system can perform reading from or writing to at least the active task variable register, the resource group use permission flag register, the resource group in-use flag register and the resource group saved value validity flag register in the above-described registers uniquely provided in the data processing apparatus.

4-2-2 Task Information and Resource Value Save Area

[0147] Also in this embodiment, to execute a plurality of tasks in parallel with each other, a task information save for storing information about each task and a resource value save area for saving resourc values indicating the status of some of the resources used in

BNSDOCID: <EP 0953905A2 I >

execution of each task are provided in a memory. The task information sav area and the resource value save area maintained for these purposes will be described in detail with reference to Fig. 10.

4-2-2-1 Task Information Save Area

[0148] Task information save areas are areas for saving task information. One set of information items corresponding to one task is stored as task information in each task information save area. Sets of task information items corresponding to tasks are collectively saved as an array in the memory according to numbering of the tasks, as are those in the above-described first embodiment of the invention. Each information set can be referred to by using the corresponding suffix in the array. That is, each task information set is saved in the memory as an array element, and the area in which each array element is saved corresponds to the task information save area. If task information sets are saved as an array in the memory in this manner, the memory can be effectively used and each task information set can be accessed in a short time.

[0149] Each task information save area includes a task control information area and a resource control information area, as shown in Fig. 10. The task control information area is an area in which task control information required to control a task is saved. Task control information saved in the task control information area is not a feature of the present invention. It may be the same as that used in the conventional systems and will not be described in detail.

[0150] The resource control information area is an area in which resource control information necessary for controlling some of the resources used by a task is saved, and which has not been provided in the conventional systems. In the resource control information area are saved a resource group monopolization flag, a resource group saved value validity flag, a resource value save area starting address, a maximum save position, and a save position array.

[0151] The resource group monopolization flag in resource control information is a flag for indication as to whether each of the resource group is monopolized by a task. This flag is a 16-bit flag such that their bits respectively correspond to the resource groups. That is, when a task is monopolizing one of the resource groups, the bit corresponding to the resource group monopolized by the task, in the bits of the resource group monopolization flag in the task information corresponding to the task, is set to the value "TRUE". Each of the bits corresponding to the resource groups not monopolized is set to the value "FALSE".

[0152] The resource group saved value validity flag in resource control information is a flag for indication as to whether some valid values have been saved in the resource value save area with respect to each of the resource groups. The resource group saved value validity flag is a 16-bit flag such that their bits respectively correspond to the resource groups. That is, if valid values have been saved in one resource value sav area as resource values of one of the resource groups, the bit corresponding to the resource group having the valid values saved, in the bits of the resource group saved value validity flag in the task information corresponding to the resource value save area, is set to the value "TRUE". Each of the bits corresponding to the resource groups having no valid values saved in the area is set to the value "FALSE".

[0153] When execution of a task is completed, saving of resource values in the resource value save area is not performed and all the bits of the resource group saved value validity flag in the task information corresponding to the task are set to the value "FALSE", thereby avoiding saving of unnecessary resource values.

[0154] A value designating the starting address of each resource value save area in the memory is set as the above-mentioned resource value save area starting address in resource control information. Conventionally, a task information save area and a resource value save area are provided as a set of adjacent areas in a memory, as shown in Fig. 20. In contrast, in this embodiment of the present invention, each task control information area and the corresponding resource value save area are maintained separately from each other in the memory. In this embodiment, therefore, the resource value save area starting address is saved as resource control information for access to each resource value save area.

[0155] As a maximum save position in resource control information, a value indicating the number of resource groups for which subareas have been secured in the corresponding resource value save area is set. For example, in the case where, as shown in Fig. 10, subareas for saving the resource values of three resource groups have been secured in one resource value save area, "3" is set as a maximum save position in the task information corresponding to the resource value save area.

[0156] As a save position array in resource control information, values designating resource value save positions are set with respect to the resource groups. For example, as shown in Fig. 10, the resource values of the resource group 0 are saved in a first location from the starting address in one resource value save area, the resource values of the resource group 8 in a second location in the resource value save area, and the resource values of the resource group 2 in a third location in the resource value save area. In this case, the value corresponding to the resource group 0 in the save position array in the task information corresponding to the resource value save area is set to the value "0" designating the stating address or the first location in the resource value sav area, the value corresponding to the resource group 8 is set to the value "1" designating

BNSDOCID: <EP 0953905A2 I > BNS page 1

the second location in the resource value save area, and the value corresponding to the resource group 2 is set to the value "2" designating the third location in the resource value save area. With respect to the resource groups having no resource values saved in the resource value save area, the value "16", which indicates that no resource values have been saved in the resource value save area, is set.

4-2-2-2 Resource Value Save Area

[0157] In each resource value save area, the resource values indicating the status of some of the resources used for execution of the task corresponding to the resource value save area are saved. However, the resource values of all the resources are not always saved in each resource value save area, and the resource values of necessary ones of the resources, collectively selected as some of the resource groups, are saved in each resource value save area.

[0158] In the conventional task switching method, the resource values of all resources are saved in the resource value save area for each of executable tasks, so that the amount of memory for all the general registers and floating-point registers, i.e., an amount of 1024 bytes, has been required as the resource value save area for one task if a data processing apparatus such as shown in Fig. 10 is employed. On the other hand, in this embodiment, resources are controlled by being separated into resource groups, and only the resource values of the resource groups to which some of the resources the values of which are to be saved belongs are stored in each resource value save area. In this embodiment, therefore, the frequency of saving resource values or the amount of saved resource values can be effectively reduced in comparison with the conventional task switching method.

[0159] In this embodiment, as described above, resources are controlled by being separated into resource groups, and only the resource values of the resource groups to which some of the resources the values of which are to be saved belong are stored in each resource value save area. Therefore, the amount of data to be saved in the memory when a task switch is made can be effectively reduced, and the memory access time required at the time of a task switch can be effectively reduced.

[0160] When the resource values of some of the resource groups are stored in one resource value save area, they are stored in unit subareas in the save area corresponding to the resource groups. At this time, the resource values of the resource groups are successively stored from the memory address designated by the resource value save area starting address in the resource control information. That is, if the resource values of, for example, the resource group 0 are to be first saved, the resource values of the resource group 0 are successively stored from the memory address designated by the resource value save area starting address in the task information. If the resource values of the resource group 8 are to be next saved, th resource values of the resource group 8 ar successively stored immediately after the subarea in which the resource values of the resource group 0 are stored. If the resource values of the resource group 2 are to be next saved, the resource values of the resource group 2 are successively stored immediately after the subarea in which the resource values of the resource group 8 are stored. Fig. 10 shows the state in which the resource values of the resource groups 0, 8, and 2 are successively stored in this manner.

4-2-3 Task Switching

[0161] An example of the task switching method using hardware such as shown in Fig. 9 will now be described in detail. "Task switching method" referred to here denotes the entire process of task switching to which the present invention is applied. That is to say, the entire process comprises three processings described below: a task switching process, a resource group usability check, and a resource group monopolizing task switching process.

[0162] If hardware such as shown in Fig. 9 is employed, either a process using the resource group use permission flag or a process not using the resource group use permission flag may be performed for task switching. Therefore, a description will be given of each of a case in which task switching is performed by using the resource group use permission flag and a case in which task switching is performed without using the resource group use permission flag.

4-2-3-1 Case of Using Resource Group Use Permission Flag

[0163] Task switching performed by using hardware such as shown in Fig. 9 and by using the resource group use permission flag will first be described.

4-2-3-1-1 Operation of Operating system when Task Switching is Performed

[0164] A process executed by the operating system to perform task switching will be described with reference to the flowchart of Fig. 11 with respect to a task switch from task A to task B.

[0165] For task switching from task A to task B, the operating system first computes in step ST4-1 the starting address (hereinafter referred to as starting address K) of the task information save area in which the task information corresponding to task B (i.e., the switched-on task) is stored. In th flowchart, "K" alone is put in to represent the starting address K.

[0166] In step ST4-2, the values of the resource group use permission flag register of the hardware are written

BNSDOCID: <EP 0953905A2 I >

BNS page 18

in the resource group monopolization flag in the task information designated by the active task variable register (i.e., task information about task A).

[0167] In step ST4-3, the values of the saved value validity flag register are written in the resource group saved value validity flag in the task information designated by the active task variable register (i.e.,,task information about task A).

[0168] In step ST4-4, the value of the starting address K computed in step ST4-1 is set in the active task variable register.

[0169] In step ST4-5, the values of the resource group monopolization flag in the task information designated by the active task variable register (i.e., task information about task B) are read and are set in the resource group use permission flag register.

[0170] In step ST4-6, the values of the resource group saved value validity flag in the task information designated by the active task variable register (i.e., task information about task B) are read and are set in the resource group saved value validity flag register.

[0171] Finally, in step ST4-7, other task switch processings are performed. After all the task switch processings have been completed, task B is started. Processings in step ST4-7 are processings other than those for resource management, and will not be described in detail because they may be the same as those performed by the conventional operating systems and do not constitute a feature of the present invention.

[0172] As described above, the process to be performed by the operating system to control resource values at the time of task switching is only rewriting of the active task variable and the flags, and the processing load is markedly small. Moreover, since no resource values are saved or read, the number of memory accessing events or the amount of data transferred by accessing the memory is markedly small and only four bytes of data is read and written. In the conventional art, all resource values are saved and read at the time of task switching and 1024 bytes of data is read and written each time task switching is performed. That is, in this embodiment of the present invention, the amount of data transferred by accessing the memory at the time of task switching is reduced to 1/256 in comparison with the conventional art.

[0173] The effect of largely reducing the processing load at the time of task switching as described above is attributed to the provision of the mechanisms for controlling resource values in the data processing apparatus. That is, it is possible to realize efficient high-speed task switching by providing the data processing apparatus with mechanisms relating to resource value control.

4-2-3-1-2 Mechanisms Provided in Hardware

[0174] In this embodiment, only rewriting of the active task variable and flags is to be executed by the operating system at the time of task switching, and saving and reading of resource values are not performed at the time of task switching. Instead, when a task is about to use one of the resources, a check is p rformed to ascertain whether the task can use the resource, and processings for saving, reading, etc., of resource values are performed if necessary to enable the task to use the resource.

[0175] In this embodiment, to achieve such a process, a mechanism for performing resource group usability check, a mechanism for performing a resource group monopolizing task switching process, a mechanism for performing a resource group saving process, a mechanism for performing a resource group reading process, and a mechanism for performing a resource group save area securing process are provided as functions of hardware.

[0176] The processes executed by the mechanisms provided as functions of hardware will now be described.

4-2-3-1-2-1 Resource Group Usability Check

[0177] Resource group usability check will first be described in detail with reference to the flowchart of Fig. 12.

[0178] A resource group usability check is executed in parallel with execution of an instruction with the central processing unit. That is, a resource group usability check is performed each time an instruction for a task is executed.

[0179] First, in step ST5-1 of the process for a resource group usability check, a determination is made as to whether an instruction to be executed for a task is a functional block transition instruction. If the instruction to be executed is a functional block transition instruction, processings in steps ST5-2 and ST5-3 are performed and the process then advances to step ST5-4. If the instruction to be executed is not a functional block transition instruction, the process moves directly to step ST5-4.

[0180] In step ST5-2, if, with respect to some of the values of the resource group in-use flag register corresponding to resource groups 0 to 3, and 8 to 11, the corresponding value of the resource group use permission flag register is "TRUE", then the value of the resource group in-use flag register is set to "FALSE". For example, if the value of the resource group use permission flag register corresponding to resource group 2 is "TRUE", the value of the resource group in-use flag register corresponding to resource group 2 is set to "FALSE". If the values of the resource group in-use flag register are set in this manner, the resource values of some of the resource groups presently monopolized by an active task with no need to hold the resource values will not be saved in the resource value save area.

[0181] In step ST5-3, the values of the resource group saved value validity flag register corresponding to resource groups 0 to 3, and 8 to 11 are set to "FALSE".

BNSDOCID: <EP 0953905A2 I > BNS page 1

If the values of the resource group saved value validity flag register are set in this manner, reading from the resource value save area will not be performed with respect to unnecessary resource values.

[0182] As described above, if an instruction to be executed is a functional block transition instruction, processings in steps ST5-2 and ST5-3 are performed to omit saving and reading of unnecessary resource values, thereby reducing the amount of data of saved or read resource values to about 50 % in comparison with the conventional art. In step ST5-4, a determination is made as to whether the instruction to be executed requires use of some of the resources. If the instruction requires use of some of the resources, the process advances to step ST5-5. If the instruction does not require use of any one of the resources, the resource group usability check is terminated and the instruction for the task is executed.

[0183] In step ST5-5, the resource group to which the resource to be used in accordance with the execution instruction belongs is identified. In the following description, the resource group identified in step ST5-5, i.e., the resource group to which the resource to be used in execution of the instruction, is referred to as "number-indexed resource group".

[0184] In step ST5-6, the resource group use permission flag corresponding to the number-indexed resource group in the resource group use permission flags stored in the resource group use permission flag register is identified. In the following description, the resource group use permission flag identified in step ST5-6 is referred to as "resource group use permission flag J". In the flowchart. "J" alone is put in to represent the resource group use permission flag J.

[0185] In step ST5-7, the resource group in-use flag corresponding to the number-indexed resource group in the resource group in-use flags stored in the resource group in-use flag register is identified. In the following description, the resource group in-use flag identified in step ST5-7 is referred to as "resource group in-use flag L". In the flowchart, "L" alone is put in to represent the resource group in-use flag L.

[0186] In step ST5-8, the resource group saved value validity flag corresponding to the number-indexed resource group in the resource group saved value validity flags stored in the resource group saved value validity flag register is identified. In the following description, the resource group saved value validity flag identified in step ST5-8 is referred to as "resource group saved value validity flag M". In the flowchart, "M" alone is put in to represent the resource group saved value validity flag M.

[0187] In step ST5-9, the value of the resource group use permission flag J is checked. If the value of the resource group use permission flag J is "FALSE", the process advances to step ST5-10. If the value of the resource group use permission flag J is "TRUE", the process moves to step ST5-12.

[0188] In step ST5-10, the resource group monopolizing task variable corresponding to the number-indexed resource group in the resource group monopolizing task variables stored in the resourc group monopolizing task variable register is identified. In the following description, the resource group monopolizing task variable identified in step ST5-10 is referred to as "resource group monopolizing task variable K". In the flowchart, "K" alone is put in to represent the resource group monopolizing task variable K.

[0189] In step ST5-11, the resource group monopolizing task switching process is performed, which will be described below in detail. After the resource group monopolizing task switching process, the process of the resource group usability check advances to step ST5-12.

[0190] In step ST5-12, the resource group use permission flag J is set to the value "TRUE", the resource group in-use flag L is set to the value "TRUE", and the resource group saved value validity flag M is set to the value "TRUE". Information that the active task is using some of the resources which belong to the number-indexed resource group is thereby recorded in the resource group in-use flag register and the resource group saved value validity flag register. After completion of the processing in step ST5-12, the process advances to step ST5-13.

[0191] In step ST5-13, a determination is made as to whether the instruction to be executed requires use of some other one of the resource groups. If the instruction requires use of another of the resource groups, the process returns to step ST5-5 and the check is performed with respect to the designated resource group. If the instruction does not require use of any one of the other resource groups, the resource group usability check is terminated and the instruction for the task is executed.

[0192] When the resource group usability check is performed as described above, the resource group monopolizing task switching process is started only if the value of the resource group usability flag J is "FALSE". If the value of the resource group usability flag J is "TRUE", the resource group monopolizing task switching process is not started. In other words, the resource group monopolizing task switching process is started in the case where, while an instruction to be executed for a task requires use of one of the resources, use of the resource is not permitted.

4-2-3-1-2-2 Resource Group Monopolizing Task Switching Process

[0193] The resource group monopolizing task switching process will be described in detail with reference to the flowchart of Fig. 13. As described above, the resource group monopolizing task switching process is a process called and started in step ST5-11 in the resource group usability check.

[0194] First, in step ST6-1 of the resource group monopolizing task switching process, the value of the resource group in-use flag L is checked. If the value of the resource group in-us flag L is "TRUE", the process advances t step ST6-2. If the value of the resource group in-use flag L is "FALSE", the process moves to step ST6-3.

[0195] In step ST6-2, the resource group saving process is performed, which will be described below in detail. After the resource group saving process, the resource group monopolizing task switching process advances to step ST6-3.

[0196] In step ST6-3, the active task variable is set to the value of the resource group monopolizing task variable K.

[0197] In step ST6-4, the value of the resource group saved value validity flag M is checked. If the value of the resource group saved value validity flag M is "TRUE", the process advances to step ST6-5. If the value of the resource group saved value validity flag M is "FALSE", the process moves to step ST6-6.

[0198] In step ST6-5, the resource group reading process is performed, which will be described below in detail. After the resource group reading process, the resource group monopolizing task switching process is finished.

[0199] In step ST6-6, the resource group save area securing process is performed, which will be described below in detail. After the resource group save area securing process, the resource group monopolizing task switching process is finished.

4-2-3-1-2-3 Resource Group Reading Process

[0200] The resource group reading process will be described in detail with reference to the flowchart of Fig. 14. As described above, the resource group reading process is a process called and started in step ST6-2 in the resource group monopolizing task switching process.

[0201] First, in step ST7-1 of the resource group reading process, the value corresponding to the number-indexed resource group in the save position array in the task information designated by the resource group monopolizing task variable K is read out. In the following description, the value of the save position array read output in step ST7-1 is referred to as "save position array value P1". In the flowchart, "P1" alone is put in to represent the save position array value P1. The save position array value P1 represents the ordinal number designating one of subarea blocks in the resource value save area in which the resource values of the number-indexed resource group will be saved.

[0202] In step ST7-2, the value of the resource value save area starting address in the task information designated by the resource group monopolizing task variable K is read out, and a value is computed which is the sum of the read starting address value and a value obtained by multiplying the save position array value P1 by 64. In the following description, the value computed in step ST7-2 is referred to as "save address Q1". In the flowchart, "Q1" alone is put in to represent the save address Q1. The save address Q1 represents the starting memory address of the subarea in which the resource values of the number-indexed resource group will be saved.

[0203] In step ST7-3, the resource values of the number-indexed resource group are saved in the subarea having the starting address corresponding to the save address Q1.

[0204] The resource group saving process is thereby completed and the resource values of the number-indexed resource group are saved in the designated location in the resource value save area.

4-2-3-1-2-4 Resource Group Reading Process

[0205] The resource group reading process will be described in detail with reference to Fig. 15. As described above, the resource group reading process is a process called and started in step ST6-5 of the resource group monopolizing task switching process.

[0206] First, in step ST8-1 of the resource value reading process, the value corresponding to the number-indexed resource group in the save position array in the task information designated by the active task variable is read out. In the following description, the value of the save position array read output in step ST8-1 is referred to as "save position array value P2". In the flowchart, "P2" alone is put in to represent the save position array value P2. The save position array value P2 represents the ordinal number designating one of subarea blocks in the resource value save area in which the resource values of the number-indexed resource group have been stored.

[0207] In step ST8-2, the value of the resource value save area starting address in the task information designated by the active task variable is read out, and a value is computed which is the sum of the read starting address value and a value obtained by multiplying the save position array value P2 by 64. In the following description, the value computed in step ST8-2 is referred to as "save address Q2". In the flowchart, "Q2" alone is put in to represent the save address Q2. The save address Q2 represents the starting memory address of the subarea in which the resource values of the number-indexed resource group have been saved.

[0208] In step ST8-3, the resource values of the number-indexed resource group are read from the subarea having the starting address corresponding to the save address Q2.

[0209] The resource group reading process is thereby completed and the resource values of the number-indexed resource group are thus read from the resource value sav area.

BNSDOCID: <EP 0953905A2 I > BNS page 2

4-2-3-1-2-5 Resource Group Save Area Securing Process

[0210] The resourc group save area securing process will be described in detail with reference to the flowchart of Fig. 16. As described above, the resource group save area securing process is a process called and started in step ST6-6 of the resource group monopolizing task switching process.

[0211] First, in step ST9-1 of the resource group save area securing process, the value corresponding to the number-indexed resource group in the save position array in the task information designated by the active task variable is read out. In the following description, the value of the save position array read output in step ST9-1 is referred to as "save position array value P3". In the flowchart, "P3" alone is put in to represent the save position array value P3.

[0212] In step ST9-2, the value of the maximum save position in the task information designated by the active task variable is read out. In the following description, the value of the maximum save position read out in step ST9-2 is referred to as "maximum save position R". In the flowchart, "R" alone is put in to represent the maximum save position R.

[0213] In step ST9-3, the save position array value P3 and the maximum save position R are compared with each other. The save position array value P3 is smaller than the maximum save position R after a subarea for saving the resource values of the number-indexed resource group has been secured in the resource value save area. Therefore, if the save position array value P3 is smaller than the maximum save position R, the resource group save area securing process is terminated.

[0214] The result of comparison in step ST9-3 that the save position array value P3 is equal to or larger than the maximum save position R corresponds to the case where no subarea for saving the resource values of the number-indexed resource group has been secured in the resource value save area. Therefore, if the save position array value P3 is equal to or larger than the maximum save position R, the resource group save area securing process is continued to secure a subarea for saving the resource values of the number-indexed resource group, and the process advances to step ST9-4.

[0215] In step ST9-4, the value of the resource value save area starting address in the task information designated by the active task variable is read out and a value is computed which is the sum of the read area and a value obtained by multiplying R+1 (the sum of maximum save position R and 1) by 64. In the following description, the value computed in step ST9-6 is referred to as "save address Q3". That is, if the value of the resource value save area starting address in the task information designated by the active task variable is A and the maximum save position R is R, save address $Q3 = A + \{(R+1) \times 64\}$. In the flowchart, "Q3" alone is put in to represent the save address Q3.

[0216] In step ST9-5, a determination is made as to whether the value of th save address Q3 is smaller than that of the starting address of a memory area used as a stack by the task (hereinafter referred to as "stack variable area"). The result of determination that the value of the save address Q3 is not smaller than that of the starting address of the stack variable area corresponds to the case where the unused area is not large enough to secure a subarea for saving the resource values of the number-indexed resource group. Therefore, if the value of the save address Q3 is not smaller than that of the starting address of the stack variable area, a stack area deficiency exception handling is started. The stack area deficiency exception handling is a processing for coping with a deficiency of the unused area. Various techniques are conceivable as means for coping with a deficiency of the unused area, and such exception handling departs from the point of the present invention. Therefore, this exception handling will not be specially explained.

[0217] The result of determination in step ST9-5 that the value of the save address Q3 is smaller than that of the starting address of the stack variable area corresponds to the case where the unused area is large enough to secure a subarea for saving the resource values of the number-indexed resource group. Therefore, if the value of the save address Q3 is smaller than that of the starting address of the stack variable area, the resource group save area securing process is continued to secure an area for saving the resource values of the number-indexed resource group, and the process advances to step ST9-6.

[0218] In step ST9-6, the value of the maximum save position (i.e., maximum save position R) in the task information designated by the active task information is written in the save position array element corresponding to the number-indexed resource group in the save position array in the task information designated by the active task variable.

[0219] In step ST9-7, the sum of the maximum save position R and 1 is written as the maximum save position in the task information designated by the active task variable. In other words, in step ST9-7, the value of the maximum save position in the task information designated by the active task variable is incremented by 1.

[0220] The resource group save area securing process is thus performed to secure an area for saving the resource values of the number-indexed resource group.

4-2-3-2 Case of Not Using Resource Group Use Permission Flag

[0221] Task switching performed by using hardware such as shown in Fig. 9 will be described with respect to a case in which task switching is performed without using the r source group use permission flag.

BNSDOCID: <EP 0953905A2 I >

4-2-3-2-1 Operation of Operating system when Task Switching is Performed

[0222] A process executed by the operating system when task switching is performed will be described with reference to the flowchart of Fig. 17 with respect to switching from task A to task B.

[0223] When task switching from task A to task B is performed, the operating system computes in step ST10-1 the starting address (hereinafter referred to as starting address K) of the task information save area in which the task information corresponding to task B (i.e., the switched-on task) is stored. In the flowchart, "K" alone is put in to represent the starting address K.

[0224] In step ST10-2, the values of the resource group saved value validity flag register are written in the resource group saved value validity flag in the task information designated by the active task variable (i.e., task information about task A).

[0225] In step ST10-3, the value of the starting address K computed in step ST10-1 is set in the active task variable register.

[0226] In step ST10-4, the values of the resource group saved value validity flag in the task information designated by the active task variable (i.e., task information about task B) are read and are set in the resource group saved value validity flag register.

[0227] Finally, other task switch processings are performed in step ST10-5, and task B is thereafter started. Processings in step ST10-5 are processings other than those for resource management, and will not be described in detail because they may be the same as those performed by the conventional operating systems and do not constitute a feature of the present invention.

[0228] As described above, the process to be performed by the operating system to control resource values at the time of task switching is only rewriting of the active task variable and the flags, and the processing load is markedly small. Moreover, since no resource values are saved or read, the number of memory accessing events or the amount of data transferred by accessing the memory is markedly small and only two bytes of data is read and written. In the conventional art, all resource values are saved and read at the time of task switching and 1024 bytes of data is read and written each time task switching is performed. That is, in this embodiment of the present invention, the amount of data transferred by accessing the memory at the time of task switching is reduced to 1/512 in comparison with the conventional art.

[0229] The effect of largely reducing the processing load at the time of task switching as described above is attributed to the provision of the mechanisms for controlling resource values in the data processing apparatus. That is, it is possible to realize efficient high-speed task switching by providing mechanisms relating to resource value control as functions of hardware.

4-2-3-2-2 Mechanisms Provided in Hardware

[0230] In this embodiment, only rewriting of th active task variable and flags is to be executed by the operating system at the time of task switching, and saving and reading of resource values are not performed at the time of task switching. Instead, when a task is about to use one of the resources, a check is performed to ascertain whether the task can use the resource, and processings for saving, reading, etc., of resource values are performed if necessary to enable the task to use the resource.

[0231] In this embodiment, to achieve such a process, a mechanism for performing resource group usability check, a mechanism for performing a resource group monopolizing task switching process, a mechanism for performing a resource group saving process, a mechanism for performing a resource group reading process, and a mechanism for performing a resource group save area securing process are provided as functions of hardware.

[0232] The processes executed by the mechanisms provided as functions of hardware will now be described. However, the resource group saving process, the resource group reading process and the resource group save area securing process in these processes are the same as those in the case of performing task switching by using the resource group use permission flag, and the description for them will not be repeated.

4-2-3-2-2-1 Resource Group Usability Check

[0233] Resource group usability check will be described in detail with reference to the flowchart of Fig. 18.

[0234] A resource group usability check is executed in parallel with execution of an instruction with the central processing unit. That is, a resource group usability check is executed each time an instruction for a task is executed.

[0235] First, in step ST11-1 of the process for a resource group usability check, a determination is made as to whether an instruction to be executed for a task is a functional block transition instruction. If the instruction to be executed is a functional block transition instruction, processing in step ST11-2 is performed and the process then advances to step ST11-3. If the instruction to be executed is not a functional block transition instruction, the process moves directly to step ST11-3.

[0236] In step ST11-2, the values of the resource group saved value validity flag register corresponding to resource groups 0 to 3, and 8 to 11 are set to "FALSE". If the values of the resource group saved value validity flag register are set in this manner, reading from the resource value save area will not be performed with respect to unnecessary resource values.

[0237] As described above, if the instruction to be exe-

BNSDOCID: <EP 0953905A2 I >

cuted is a functional block transition instruction, processing in step ST11-2 is performed to omit saving and reading of unnecessary resource values, thereby reducing the amount of saved or read resource values to about 50 % in comparison with the conventional art.

[0238] In step ST11-3, a determination is made as to whether the instruction to be executed requires use of some of the resources. If the instruction requires use of some of the resources, the process advances to step ST11-4. If the instruction does not require any one of the resources, the resource group usability check is terminated and the instruction for the task is executed.

[0239] In step ST11-4, the resource group to which the resource to be used in accordance with the instruction to be executed belongs is identified. In the following description, the resource group identified in step ST11-4, i.e., the resource group to which the resource to be used in execution of the instruction belongs, is referred to as "number-indexed resource group".

[0240] In step ST11-5, the resource group monopolizing task variable corresponding to the number-indexed resource group in the resource group monopolizing task variables stored in the resource group monopolizing task variable register is identified. In the following description, the resource group monopolizing task variable identified in step ST11-5 is referred to as "resource group monopolizing task variable K". In the flowchart, "K" alone is put in to represent the resource group monopolising task variable K.

[0241] In step ST11-6, the resource group saved value validity flag corresponding to the number-indexed resource group in the resource group saved value validity flags stored in the resource group saved value validity flag register is identified. In the following description, the resource group saved value validity flag identified in step ST11-6 is referred to as "resource group saved value validity flag M". In the flowchart, "M" alone is put in to represent the resource group saved value validity flag M.

[0242] In step ST11-7, the active task variable and the resource group monopolizing task variable K are compared. In the flowchart, "D" represents the active task variable. If the value of the active task variable and the value of the resource group monopolizing task variable K are not equal to each other, the process advances to step ST11-8. If they are equal to each other, the process moves to step ST11-9.

[0243] In step ST11-8, the resource group monopolizing task switching process is performed, which will be described below in detail. After the resource group monopolizing task switching process, the process of the resource group usability check advances to step ST11-9.

[0244] In step ST11-9, the value of the resource group saved value validity flag M is set to "TRUE". Information that conditi ns for saving of the resource values of the number-indexed resource group have been satisfied is recorded in the resourc group saved value validity flag register. After completion of the processing in step ST11-9, the process advances to step ST11-10.

[0245] In step ST11-10, a determination is made as to whether th instruction to be executed requires use of some ther one of the resource groups. If the instruction requires use of another of the resource groups, the process returns to step ST11-4 and the check is performed with respect to the designated resource group. If the instruction does not require use of any one of the other resource groups, the resource group usability check is terminated and the instruction for the task is executed.

[0246] When the resource group usability check is performed as described above, the resource group monopolizing task switching process is started only if the value of the active task variable and the value of the resource group monopolizing task variable K are not equal to each other. If the value of the active task variable and the value of the resource group monopolizing task variable K are equal to each other, the resource group monopolizing task switching process is not started. In other words, the resource group monopolizing task switching process is started in the case where, while an instruction to be executed for a task requires use of one of the resources, the task is not monopolizing the resource.

4-2-3-2-2-2 Resource Group Monopolizing Task Switching Process

[0247] The resource group monopolizing task switching process will be described in detail with reference to the flowchart of Fig. 19. As described above, the resource group monopolizing task switching process is a process called and started in step ST11-9 in the resource group usability check.

[0248] First, in step ST12-1 of the resource group monopolizing task switching process, the value of the resource group saved value validity flag corresponding to the number-indexed resource group in the resource group saved value validity flags in the task information designated by the resource group monopolizing task variable K is read out. In the following description, the value of the resource group saved value validity flag read out in step ST12-1 is referred to as "resource group saved value validity flag L". In the flowchart, "L" alone is put in to represent the resource group saved value validity flag L.

[0249] In step ST12-2, the value of the resource group saved value validity flag L is checked. If the value of the resource group saved value validity flag L is "TRUE", the process advances to step ST12-3. If the value of the resource group saved value validity flag L is "FALSE", the process moves to step ST12-4.

[0250] In step ST12-3, the resource group saving process is performed in accordance with the flowchart of Fig. 14. After the resource group saving process, the resource group monopolizing task switching process

BNSDOCID: <EP 0953905A2 I >

advances to step ST12-4.

[0251] In step ST12-4, the active task variable is set to the valu of resource group monopolising task variable K.

[0252] In step ST12-5, the value of th resource group saved value validity flag M is checked. If the value of the resource group saved value validity flag M is "TRUE", the process advances to step ST12-6. If the value of the resource group saved value validity flag M is "FALSE", the process moves to step ST12-7.

[0253] In step ST12-6, the resource group reading process is performed in accordance with the flowchart of Fig. 15. After the resource group reading process, the resource group monopolizing task switching process is finished.

[0254] In step ST12-7, the resource group save area securing process is performed in accordance with the flowchart of Fig. 16. After the resource group save area securing process, the resource group monopolizing task switching process is finished.

4-2-3-3 Recapitulation

[0255] As described above, in each of the case of using the resource group use permission flag and the case of not using the resource group use permission flag, the process to be performed by the operating system to control resource values at the time of task switching is only rewriting of the active task variable and the flags, and the processing load is markedly small. Moreover, since no resource values are saved or read at the time of task switching, the number of memory accessing events or the amount of data transferred by accessing the memory is markedly small. That is, mechanisms relating to control of resource values are provided as functions of hardware to perform saving and reading of resource values by hardware processing instead of processing based on the operating system alone. In this manner, saving and reading of resource values required when tasks are switched can be efficiently performed at a high-speed.

[0256] In the above-described task switching process, saving and reading of resource values are not simply postponed but completely omitted with respect to resource groups not used by a started task. Also, with respect to used resource groups, their values which have been set during use are maintained if they will not be used by other tasks. In such a case, saving and reading of the resource values are completely omitted. During execution of a task, each time a functional block transition instruction is executed, the resource group in-use flag register and the resource group saved value validity flag register are updated and saving is not performed with respect to resource groups with no need to save the values thereof. As described above, the number of resource value saving and reading events or the amount of saved or read resource values is minimized and only necessary ones of resource values are saved or read. Thus, saving and reading of resource values can be performed remarkably efficiently.

[0257] In the conventional task switching method, there is a possibility of loads for reading from a memory and writing to the memory being concentrated at th time of switching to cause a bottleneck. However, if task switching is performed in the above-described manner, saving and reading of resource values are sporadically performed as described above, so that a bottle neck cannot occur easily.

[0258] Moreover, if task switching is performed in the above-described manner, the amount of resource value data which needs saving or reading is markedly small, so that task switching can be performed at a markedly high speed. Therefore, a processing which needs to start quickly, e.g., interrupt processing can be perform by using the task switching mechanism. In other words, if task switching is performed in the above-described manner, the overhead at a start or end of interrupt processing performed by using the task switching mechanism is advantageously small.

5. Other Embodiments of the Invention

[0259] The present invention is not limited to the above-described embodiment and can be modified in various ways.

[0260] For example, the number of registers for the central processing unit and the number of registers for the floating-point arithmetic unit are not specially limited and may be set to any number selected as desired. Further, the hardware of the present invention is not limited to the above-described one formed by the central processing unit and the floating-point arithmetic unit. For example, the hardware may be a central processing unit alone or may include various coprocessors other than the floating-point arithmetic unit, e.g., a unit for performing a calculation for three-dimensional coordinate conversion, and a unit for performing parallel operations for high-speed image data compression/expansion processing.

[0261] When resources usable by tasks are separated into resource groups to be controlled, the number of groups are not particularly limited and may be freely selected. However, the efficiency of task switching varies depending upon the way of grouping resources. Therefore, it is, of course, preferable that, in grouping and controlling resources, grouping of the resources is optimized according to the sorts of tasks to be executed, the configuration of the data processing apparatus, etc., so that task switching can be performed more efficiently. If the central processing unit has a cache memory as mentioned above, it is preferred that resources are separated into resource groups in such a manner that the size of the resource value save area corresponding to each resource group is an integer multiple of the cache line size of the cache memory.

[0262] In th second mbodiment of the present

invention, almost all the processes other than the task switching process are realized through mechanisms in hardwar . However, almost entire portions of th hardware mechanisms may be arbitrarily selected to be replaced with softwar according to the required scale of a semiconductor circuit, etc. In such a case, the hardware and software configuration may be such that, while the resource group monopolizing task switching process is performed by software, some of the functions for the other processes are extracted to be realized by means for hardware. For example, special short sequences of instructions may be prepared to enable hardware to perform high-speed processing for saving and reading of resource groups.

[0263] In the second embodiment of the present invention, the starting address of the task information save area corresponding to an active task is set as the value of the active task variable. Alternatively, the starting address of the resource control information area in the task information area corresponding to an active task may be set as the value of active task variable. If the starting address of the resource control information area in the task information area is set as the active task variable, the resource control information area can be directly accessed based on the active task variable with no influence of the portion of task information depending upon the operating system (i.e., task control information). In other words, if the starting address of the resource control information area in task information is set as the active task variable, management of task information necessary for resource control can be performed only by hardware processing without being influenced by the configuration of the operating system.

[0264] Finally, the configuration of an example of hardware to which the present invention is applied is described below with reference to Fig. 22. Fig. 22 illustrates an example of an application of the present invention to a television receiver 1. Needless to say, the present invention can be applied to various information processing apparatuses other than the television receiver 1. That is, the present invention can be applied to a wide variety of information processing apparatuses having operating systems (hereinafter referred to as "OS") each of which manages a plurality of tasks and requires task switching. For example, the present invention can be applied to various audio-visual apparatuses, information processing apparatuses for office work, and computer-controlled machines.

[0265] Referring to Fig. 22, the television receiver 1 is an apparatus which receives, via an antenna or a cable, a signal transmitted from a broadcasting station, displays an image on an image display unit based on the received signal, and outputs audio information through a speaker.

[0266] The television receiver 1 has, as well as an rdinary television receiving functi n, a function for receiving a program, data, etc., externally supplied. As shown in Fig. 22, the television receiver 1 has components connected to a bus 3: a processor 6 connected to the bus 3 through a bus/memory bridge 5; a read-only memory (ROM) 7 and a random access memory (RAM) 8 connected t the bus 3 thr ugh the bus/memory bridge 5; an operating panel 9; an external storage unit 10; and a communication unit 11.

[0267] A television function section 4 has a function of reproducing images and sounds based on signals received via an antenna, a cable or the like. The television function section 4 is connected to the bus 3 through a bus/IO bridge 2 to be able to exchange signals with the other sections.

[0268] The processor 6 is adapted to control each section of this television receiver 1, and is connected to the bus 3 through the bus/memory bridge 5. The ROM 7 and the RAM 8 are connected to the processor 6 through the bus/memory bridge 5. The ROM 7 holds, for example, fixed information about the television receiver 1. The RAM 8 provides a memory space to be used as a work area by the processor 6. That is, the processor 6 controls each section constituting this television receiver 1 by executing the operation of an OS (including the task switching apparatus and method of the present invention) and an application program recorded in the external storage unit 10 while using the RAM 8 as a work area, as described below.

[0269] The operating panel 9 is an input/output device for enabling user's input operations. For example, a signal designating selection of a channel or a change in sound volume is input to the television receiver 1 through the operating panel 9. Specifically, the operating panel 9 is constituted by an input device having a plurality of switches for inputting various signals, a pointing device represented by a so-called mouse, etc. A signal input by the operating panel 9 is input to the processor 6 via the bus 3 and the bus/memory bridge 5. The processor 6 then performs predetermined arithmetic processing based on the signal input from the operating panel 9 to control the corresponding section of the television receiver 1.

[0270] The external storage unit 10 is constituted by a storage unit such as a hard disk unit, for example. The external storage unit 10 has the OS (including the task switching apparatus and method of the present invention), an application program, etc., recorded therein . In the external storage unit 10, image data, control data, a program externally downloaded through the communication unit 11, etc., are also recorded. The communication unit 11 is an input/output section for performing data communication with an external terminal. For example, the communication unit 11 is constituted by a communication device such as a modem or a terminal adapter.

[0271] In the television receiver 1, to control each section, the processor 6 executes the operation of the OS recorded in the external storage unit 10, and executes an application program recorded in the external storage unit 10 by using, on the OS, control information or the

BNSDOCID: <EP 0953905A2 I >

like recorded on the ROM 7. That is, this television receiver 1 has the external storage unit 10 used as a program provision medium for providing various data processing programs forming the OS.

[0272] The OS including the task switching apparatus and method of the present invention and may be recorded in the ROM 7 or RAM 8. In such a case, the ROM 7 or RAM 8 is provided as a program provision medium. If version-up rewriting or the like of the OS is taken into consideration, it is desirable to use a rewritable recording medium for provision of the OS including the task switching apparatus and method of the present invention. The program provision medium in accordance with the present invention may be, for example, a magneto-optic recording medium or an optical recording medium used by being detachably loaded in the television receiver 1, or a network circuit or the like for connecting the television receiver 1 to other various information processing apparatuses.

[0273] The OS provided in the television receiver 1 is a pure object-oriented OS. On the OS, an application program for displaying moving images through the television function section 4, and an application program for realizing a graphical user interface (GUI) for controlling the operating panel 9 are executed.

[0274] As described above in detail, according to the present invention, the number of memory accessing events at the time of task switching can be reduced, the number of times the memory is unnecessarily accessed can be effectively reduced, and the need to a large amount of hardware resources for holding resource values can be eliminated. At least preferred embodiments of the present invention, therefore, have the following advantages:

(1) In a central processing unit having a large number of registers, the time required for switching tasks can be largely reduced.

(2) In a case where a floating-point arithmetic unit, for example, is added for hardware function expansion, the task switching time is not increased with respect to tasks not using the added function.

(3) Even if the amount of usable resources is increased, there is no need for an extra memory space other than the memory area for saving resources of resource groups used by tasks, and the memory consumption is advantageously limited.

(4) The effect of the present invention is not lowered by an increase in the number of tasks to be switched, and task switching can be performed in a short time.

(5) The processing apparatus of the present invention requires only additional hardware with respect to a large amount of resources for the central processing unit, etc. Therefore, the semiconductor ar a can be largely reduced in comparison with the case of achieving high-speed task switching based on a register file mechanism or the like.

(6) Even if the amounts of resources for the central processing unit, etc., is increased, the amount of additional hardware required by the present invention is small and the number of tasks switchable at a high speed is not particularly limited.

(7) For tasks requiring a quick start, the task switching speed can be increased by limiting the amount of usable resources. Even in the case of an interrupt processing or the like requiring a high-speed start, the desired effect can be achieved by using a suitable task switching mechanism.

(8) Memory accessing events before completion of task switching can be effectively reduced, so that the start time of a switched-on task can easily be specified.

(9) If a resource use inhibition function (e.g., the floating-point unit use permission flag) is prepared as a hardware function, speedup of task switching can be achieved only by a change in the operating system.

(10) If mechanisms for performing a resource group usability check and a resource group monopolizing task switching process are provided as hardware functions in a data processing apparatus , the number of instruction steps necessary for task switching can be reduced to increase the task switching processing speed. Moreover, the speed of monopolizing task switching started when resources are first used after task switching can be largely increased.

(11) It is also possible to prevent occurrence of unnecessary memory accessing events at starting and trailing ends of the resource value save area by setting the sizes of resource groups such that they correspond to an integer multiple of the cache line size of a cache memory, or by adjusting the starting address of resource value save subareas to a cache line boundary. The memory access efficiency can be improved thereby.

Claims

1. A task switching apparatus in a multitasking system comprising:

a plurality of tasks executed in parallel with each other while independence of each task is ensured, each task having a resource value save area for its exclusive use;

a plurality of resource groups for controlling resources in a separate state;

a task switcher which switches the tasks so that a second one of the tasks becomes active in place of a first one of the tasks;

a resourc group usability checker for checking, when the active task uses one-of the resources which b long to one of the plurality

BNSDOCID: <EP 0953905A2 I >

of groups, whether the active task has a right to use the resource group to which the resource to b used by the active task belongs; and

a resource group monopolizing task switcher for giving the active task a right to use the resource group to which the resource to be used by the active task belongs if the active task has no right to use the resource group.

2. A task switching apparatus according to Claim 1, wherein, if there is a possibility of said resource group usability checker being unable to check a right to use with respect to one of the plurality of resource groups, said task switcher comprises:

means for saving values of the resources of said one of the resource groups in the resource value save area for the first task; and

means for reading, to the resources of said one of the resource groups, values saved in the resource value save area for the second task.

3. A task switching apparatus according to Claim 1, wherein each of the plurality of tasks has a resource group use permission flag for indication as to whether the task has a right to use one of the resource groups, and said resource group usability checker has means for making a determination as to whether the active task has a right to use one of the resource groups based on a value of the resource group use permission flag.

4. A task switching apparatus according to Claim 3, wherein each of the plurality of tasks has, in one-to-one correspondence with the plurality of resource groups, resource group monopolization flags for indication as to whether the task monopolizes at least one of the plurality of resource groups, said task switcher having:

means for saving the value of the resource group use permission flag as a value of the corresponding resource group monopolization flag of the first task; and

means for reading the value of each of the resource group monopolization flags of the second task to the corresponding resource group use permission flag, and said resource group monopolizing task switcher having:

means for setting a value indicating that one of the tasks having a right to use one of the resource groups does not monopolize said one of the resource groups if said one of the resource groups is the resource group to which the resource that the active task needs to have a right to use belongs, the value being set in the resource group monopolization flag corresponding to said one of the resource groups in the resource group monopolization flags of said one of the tasks having a right to use said one of the resource groups.

5. A task switching apparatus according to Claim 3, wherein the resource groups are separated into functional blocks each containing at least one of the resource groups providing a common function, and each of the functional blocks has the resource group use permission flag for indication as to whether the active task has a right to use the functional block.

6. A task switching apparatus according to Claim 3, wherein each of the plurality of resource groups has the resource group use permission flag for indication as to whether the active task has a right to use the resource group.

7. A task switching apparatus according to Claim 2, wherein said resource group monopolizing task switcher has:

means for saving, in the resource value save area for the active task, values of the resource group to which the resource that the active task is about to use belongs if there is a need to save at least one resource value of the resource; and

means for reading any valid value saved in the resource value save area for the active task as a resource value of the resource that the active task is about to use, the resource value being read to the resource value save area for the active task.

8. A task switching apparatus according to Claim 7, wherein resource group monopolizing task variables each set to a value designating one of the tasks monopolizing one of the plurality of resource groups are provided, and said resource group monopolizing task switcher has means for setting a value designating the active task as the resource group monopolizing task variable corresponding to the resource group to which the resource that the active task is about to use belongs.

9. A task switching apparatus according to Claim 8, wherein an active task variable which is set to a value designating the active task is provided, and said resource group usability checker makes a determination as to whether the active task can use the resource group to which the resource that the active task is about to use belongs by making a comparison between the value of the active task variable and the value of the resource group monopolizing task variable corresponding to the resource group to which the resource that the

active task is about to use belongs.

10. A task switching apparatus according to Claim 7, wherein each of the plurality of tasks has, in one-to-one correspondence with th plurality of resource groups, resource group saved value validity flags for indication as to whether the task is using at least one of the plurality of resource groups, said resource group monopolizing task switcher having:

means for setting a value indicating that one of the resources in one of the resource groups has been used by the active task in the resource group saved value validity flag corresponding to said one of the resource groups to which said one of the resources belongs in the resource group saved value validity flags corresponding to the active task when at least one value of said one of the resources becomes effective as a result of use of said one of the resources by the active task;

means for making a determination as to whether there is a need to save at least one resource value of the resource that the active task is about to use, the determination being made based on the value of the resource group saved value validity flag corresponding to one of the tasks using the resource that the active task is about to use; and

means for making a determination as to whether at least one valid value has been saved in the resource value save area for the active task as a resource value of the resource that the active task is about to use, the determination being made based on the value of the resource group saved value validity flag corresponding to the active task.

11. A task switching apparatus according to Claim 10, wherein, if there is a possibility of said resource group usability checker being unable to check a right to use with respect to one of the plurality of resource groups, said task switcher comprises means for setting, in the resource group saved value validity flag corresponding to said one of the resource groups, a value indicating that at least one valid value exists in the resource value save area corresponding to said one of the resource groups.

12. A task switching apparatus according to Claim 10, further comprising means for setting, before execution of one of the tasks is stopped, the value of each of the resource group saved value validity flags corresponding to the task to a value indicating that none of the resources in the resource group is in use.

13. A task switching apparatus according to Claim 7, wherein each of said plurality of tasks has resource group in-use flags in one-to-one correspondence with the plurality of resource groups, said resource group monopolizing task switcher having:

means for setting, in the resource group in-use flag corresponding to the resource group to which one of the resources that one of the tasks uses belongs, a value indicating that said one of the tasks is using the resource that belongs to said one of the resource groups; and

means for making a determination as to whether there is a need to save a resource value of the resource that the active task is about to use, the determination being made based on the value of the resource group in-use flag corresponding to the resource group to which the resource that the active task is about to use belongs.

14. A task switching apparatus according to Claim 13, wherein, if there is a possibility of said resource group usability checker being unable to check a right to use with respect to one of the plurality of resource groups, said task switcher comprises means for setting, in the resource group in-use flag corresponding to said one of the resource groups, a value indicating that at least one of the resources that belong to said one of the resource groups is in use.

15. A task switching apparatus according to Claim 7, wherein each of said plurality of tasks has a resource group monopolization flag for indication as to whether the task is monopolizing each of the plurality of resource groups, said resource group monopolizing task switcher having:

means for setting a value indicating that one of the resource groups is not being monopolized if one of said resource groups is the resource group to which the resource that the active task is about to use belongs, the value being set in the resource group monopolization flag corresponding to the resource group to which the resource that the active task is about to use belongs in the resource group monopolization flags of one of the tasks monopolizing said one of the resource groups; and

means for setting a value indicating that the resource group to which the resource that the active task is about to use belongs is being monopolized, the value being set in the resource group monopolization flag corresponding to the resource groups to which the resource that the active task is about to use belongs in the resource gr up monopolization

BNSDOCID: <EP 0953905A2 I >

BNS page 2

flags corresponding to the active task.

16. A task switching apparatus according to Claim 1, wherein the resources that belong to th resource groups are processed with a central processing unit having a cache memory, the task switching apparatus further comprising means for dividing each of the resource value save areas into subareas corresponding to some of the resource groups so that each subarea has a size corresponding to an integer multiple of a cache line size of the cache memory.

17. A task switching apparatus according to Claim 1, wherein the resource value save areas are secured on a cache memory, the task switching apparatus further comprising means for securing the resource value save areas so that the starting address of each resource value save area corresponds to a cache line boundary.

18. A task switching apparatus according to Claim 1, further comprising means for securing the resource value save areas while dynamically changing the size of each resource value save area according to the size of saved resource values.

19. A task switching apparatus according to Claim 18, further comprising means for dynamically changing the size of the resource value save area with respect to each of the resource groups.

20. A task switching apparatus according to Claim 18, further comprising means for securing each of the resource value save areas in one memory block in which a stack used for the corresponding task is also provided, the resource value save area being secured in a direction opposite from the direction in which data is stored in the stack by the task.

21. A task switching apparatus according to Claim 20, further comprising means for checking whether the resource value save area and the area used as the stack by the task do not overlap each other when the resource value save area is dynamically changed.

22. A task switching method in a multitasking system comprising:

an execution step of executing a plurality of tasks in parallel with each other while independence of each task is ensured, each task having a resource value save area for its exclusive use;
a storing step of storing a plurality of resource groups in a separate state;
a task switching step of switching the tasks so that a second one of the tasks becomes active in place of a first one of the tasks;
a resource group usability checking step of checking, when the active task uses one of the resources which belong to ne of the plurality of groups, whether the active task has a right to use the resource group to which the resource to be used by the active task belongs; and
a resource group monopolizing task switching step of giving the active task a right to use the resource group to which the resource to be used by the active task belongs if the active task has no right to use the resource group.

23. A task switching method according to Claim 22, wherein, if there is a possibility of said resource group usability checking step being ineffective in checking a right to use with respect to one of the plurality of resource groups, said task switching step includes:

a step of saving values of the resources of said one of the resource groups in the resource value save area for the first task; and
a step of reading, to the resources of said one of the resource groups, values saved in the resource value save area for the second task.

24. A task switching method according to Claim 22, wherein said resource group usability checking step includes a step of making a determination as to whether the active task has a right to use the resource group to which the resource that the active task is about to use belongs based on a value of a resource group use permission flag corresponding to the resource group.

25. A task switching method according to Claim 24, wherein a resource group use permission flag is provided with respect to each of functional blocks formed at least one of the resource groups providing a common function, and said resource group usability checking step includes a step of making a determination as to whether the active task has a right to use the functional block to which the resource that the active task is about to use belongs based on a value of the corresponding resource group use permission flag.

26. A task switching method according to Claim 24, wherein said task switching step includes:

a step of saving, in each of resource group monopolization flags of the first task, a value of a corresponding resource group use permission flag; and
a step of reading a value of each of resource group monopolization flags of the second task

BNSDOCID: <EP 0953905A2 I >

to a corresponding resource group use permission flag, and said resourc group monopolizing task switching step includes:

a step of setting a value indicating that one of the tasks having a right to use one of th resource groups does not monopolize said one of the resource groups if said one of the resource groups is the resource group to which the resource that the active task is about to use belongs, the value being set in the resource group monopolization flag corresponding to said one of the resource groups in the resource group monopolization flags of said one of the tasks having a right to use said one of the resource groups.

27. A task switching method according to Claim 23, wherein said resource group monopolizing task switching step includes:

a step of saving, in the resource value save area for the active task, values of the resource group to which the resource that the active task is about to use belongs if there is a need to save at least one resource value of the resource; and

a step of reading any valid value saved in the resource value save area for the active task as a resource value of the resource that the active task is about to use, the resource value being read to the resource value save area for the active task.

28. A task switching method according to Claim 27, wherein said resource group monopolizing task switching step includes a step of setting a value designating the active task as a resource group monopolizing task variable corresponding to the resource group to which the resource that the active task is about to use belongs.

29. A task switching method according to Claim 28, wherein said resource group usability checking step includes a step of making a determination as to whether the active task can use the resource group to which the resource that the active task is about to use belongs by making a comparison between the value of an active task variable designating the active task and the value of the resource group monopolizing task variable corresponding to the resource group to which the resource that the active task is about to use belongs.

30. A task switching method according to Claim 27, wherein said resource group monopolizing task switching step includes:

a step of setting a valu indicating that one of the resources in one of the resource groups has been used by the active task in a resource group saved value validity flag corresponding to said one of the resourc groups to which said one of the r sources belongs in resource group saved value validity flags corresponding to the active task when at least one value of said one of the resources becomes effective as a result of use of said one of the resources by the active task;

a step of making a determination as to whether there is a need to save at least one resource value of the resource that the active task is about to use, the determination being made based on a value of a resource group saved value validity flag corresponding to one of the tasks using the resource that the active task is about to use; and

a step of making a determination as to whether at least one valid value has been saved in the resource value save area for the active task as a resource value of the resource that the active task is about to use, the determination being made based on the value of the resource group saved value validity flag corresponding to the active task.

31. A task switching method according to Claim 30, wherein, if there is a possibility of said resource group usability checking step being ineffective in checking a right to use with respect to one of the plurality of resource groups, said task switching step includes a step of setting, in a resource group saved value validity flag corresponding to said one of the resource groups, a value indicating that at least one valid value exists in the resource value save area corresponding to said one of the resource groups.

32. A task switching method according to Claim 30, further comprising a step of setting, before execution of one of the tasks is stopped, the value of each of the resource group saved value validity flags corresponding to the task to a value indicating that none of the resources in the resource group is in use.

33. A task switching method according to Claim 27, wherein said resource group monopolizing task switching step includes:

a step of setting, in a resource group in-use flag corresponding to the resource group to which one of the resources that one of the tasks uses belongs, a value indicating that said one of the tasks is using the resource that belongs to said one of the resource groups; and

a step of making a determination as to whether there is a need to save a resource value of the

BNSDOCID: <EP 0953905A2 1 >

resource that the active task is about to use, the determination being made based on a value of a resource group in-us flag corresponding to the resource group to which the resource that the active task is about to use belongs.

34. A task switching method according to Claim 33, wherein, if there is a possibility of said resource group usability checking step being ineffective in checking a right to use with respect to one of the plurality of resource groups, said task switching step includes a step of setting, in a resource group in-use flag corresponding to said one of the resource groups, a value indicating that at least one of the resources that belong to said one of the resource groups is in use.

35. A task switching method according to Claim 27, wherein said resource group monopolizing task switching step include:

means for setting a value indicating that one of the resource groups is not being monopolized if said one of the resource groups is the resource group to which the resource that the active task is about to use belongs, the value being set in a resource group monopolization flag corresponding to the resource group to which the resource that the active task is about to use belongs in resource group monopolization flags corresponding to one of the tasks monopolizing said one of the resource groups; and
a step of setting a value indicating that the resource group to which the resource that the active task is about to use belongs is being monopolized, the value being set in a resource group monopolization flag corresponding to the resource group to which the resource that the active task is about to use belongs in resource group monopolization flags corresponding to the active task.

36. A task switching method according to Claim 22, wherein the resources that belong to the resource groups are processed with a central processing unit having a cache memory, the task switching method further comprising a step of dividing each of the resource value save areas into subareas corresponding to some of the resource groups so that each subarea has a size corresponding to an integer multiple of a cache line size of the cache memory.

37. A task switching method according to Claim 22, wherein the resource value save areas are secured on a cache memory, the task switching method further comprising a step of securing the resourc value save areas so that the starting address of each resource value save area corresponds to a cache line boundary.

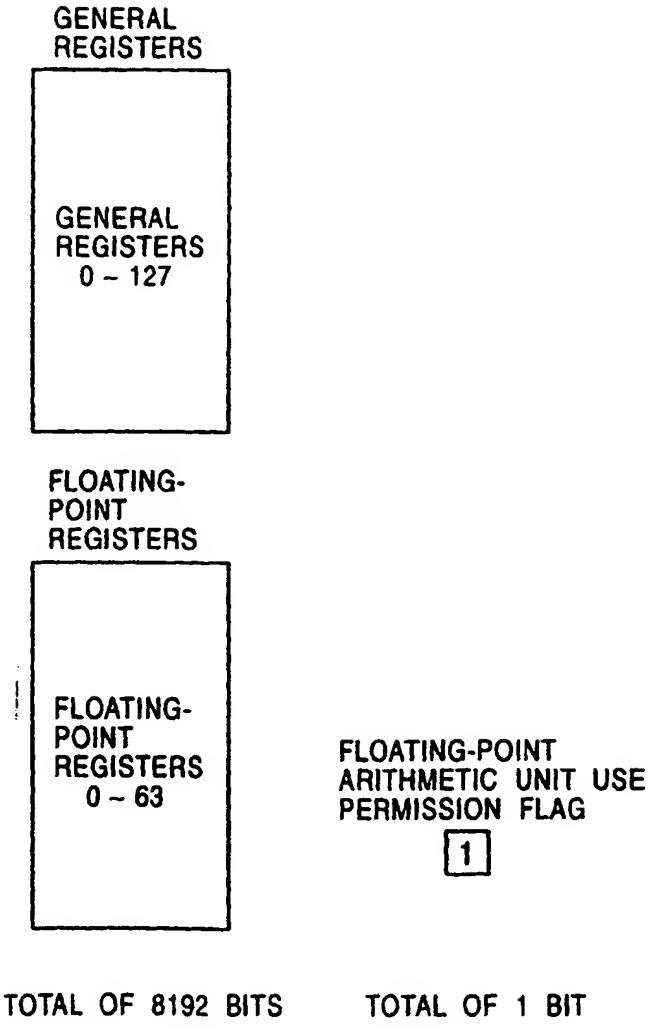
38. A task switching method according to Claim 22, further comprising a step of securing the resource value save areas while dynamically changing the size of each resource value save area according to the size of saved resource values.

39. A task switching method according to Claim 38, further comprising a step of dynamically changing the size of the resource value save area with respect to each of the resource groups.

40. A task switching method according to Claim 38, further comprising a step of securing each of the resource value save areas in one memory block in which a stack used for the corresponding task is also provided, the resource value save area being secured in a direction opposite from the direction in which data is stored in the stack by the task.

41. A task switching method according to Claim 40, further comprising a step of checking whether the resource value save area and the area used as the stack by the task do not overlap each other when the resource value save area is dynamically changed.

FIG. 1



BNSDOCID: <EP 0953905A2 I >

BNS page 3

FIG. 2

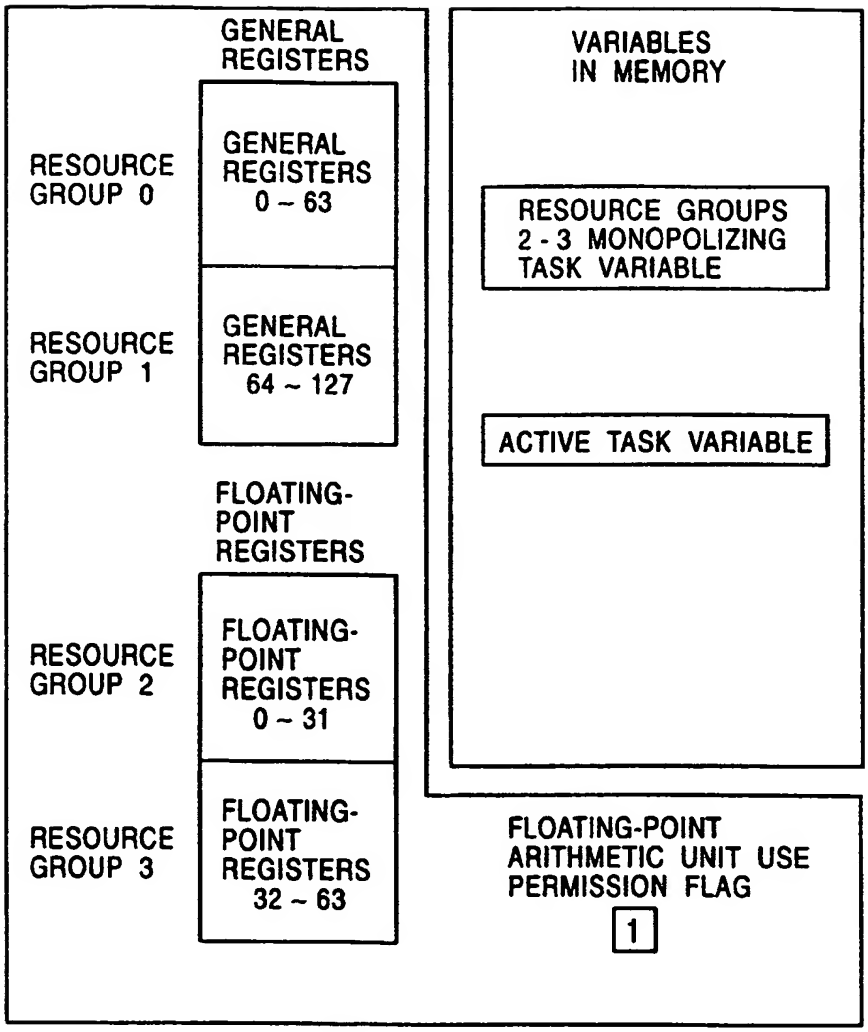
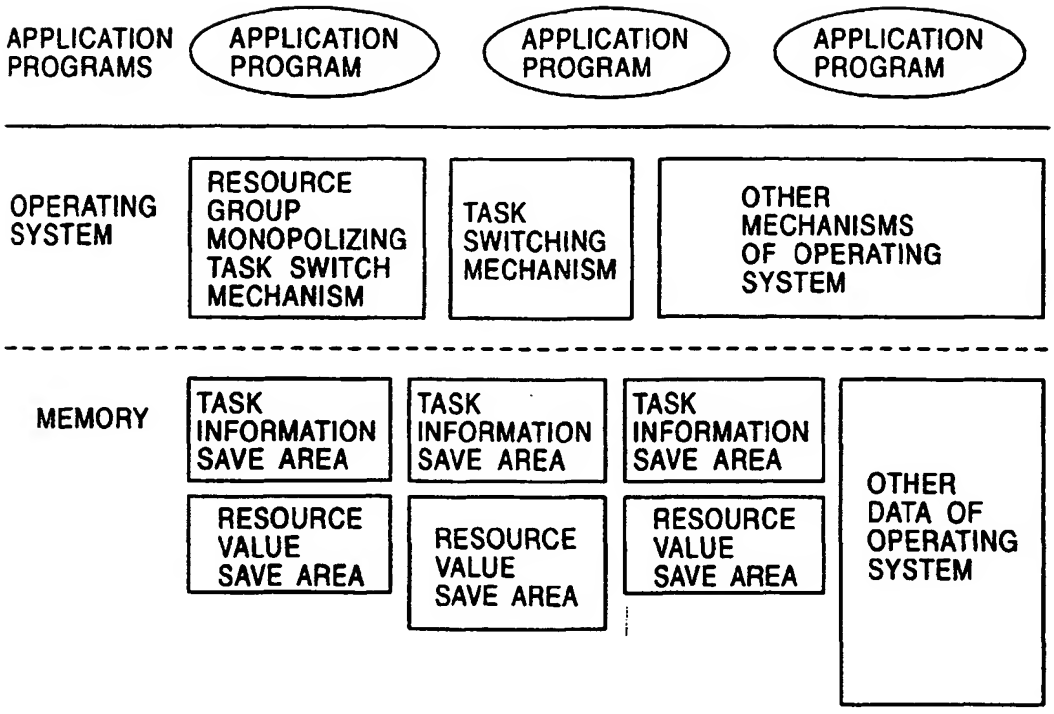

GENERAL REGISTERS
RESOURCE GROUP 0
GENERAL REGISTERS 0 ~ 63
RESOURCE GROUP 1
GENERAL REGISTERS 64 ~ 127
FLOATING-POINT REGISTERS
RESOURCE GROUP 2
FLOATING-POINT REGISTERS 0 ~ 31
RESOURCE GROUP 3
FLOATING-POINT REGISTERS 32 ~ 63
VARIABLES IN MEMORY
RESOURCE GROUPS 2 - 3 MONOPOLIZING TASK VARIABLE
ACTIVE TASK VARIABLE
FLOATING-POINT ARITHMETIC UNIT USE PERMISSION FLAG
1

FIG. 3



BNSDOCID: <EP 0953905A2 I >

FIG. 4

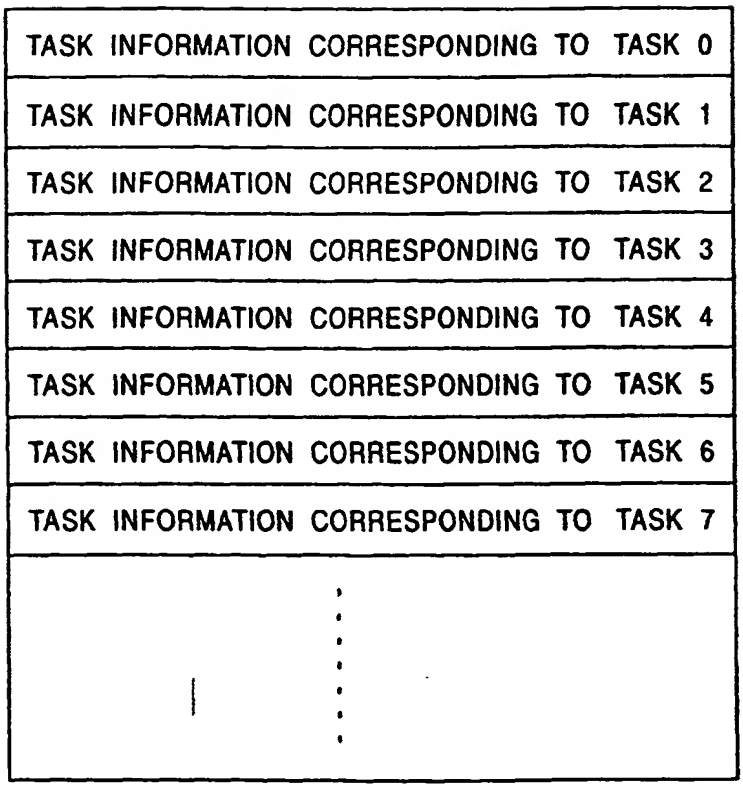
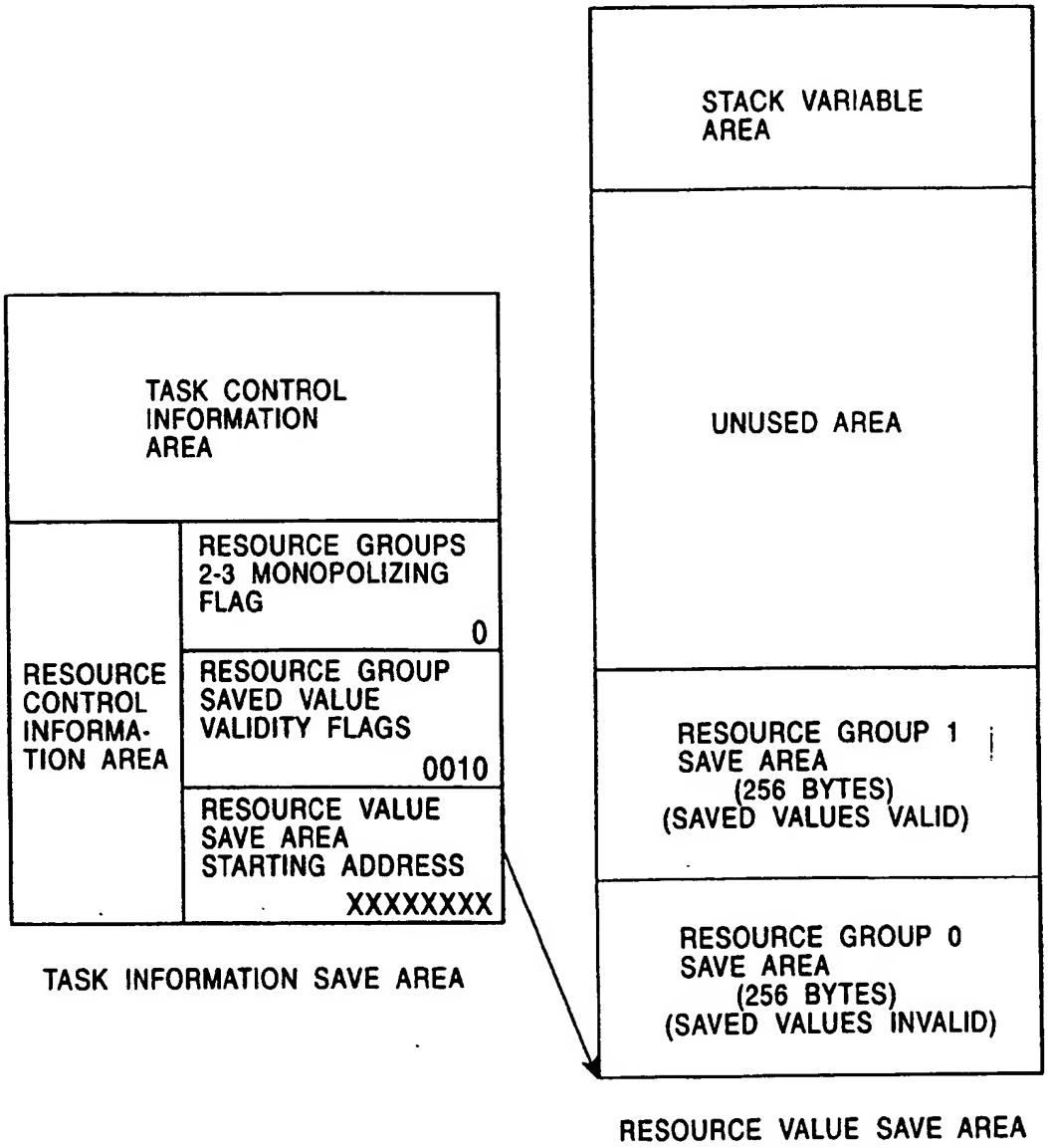
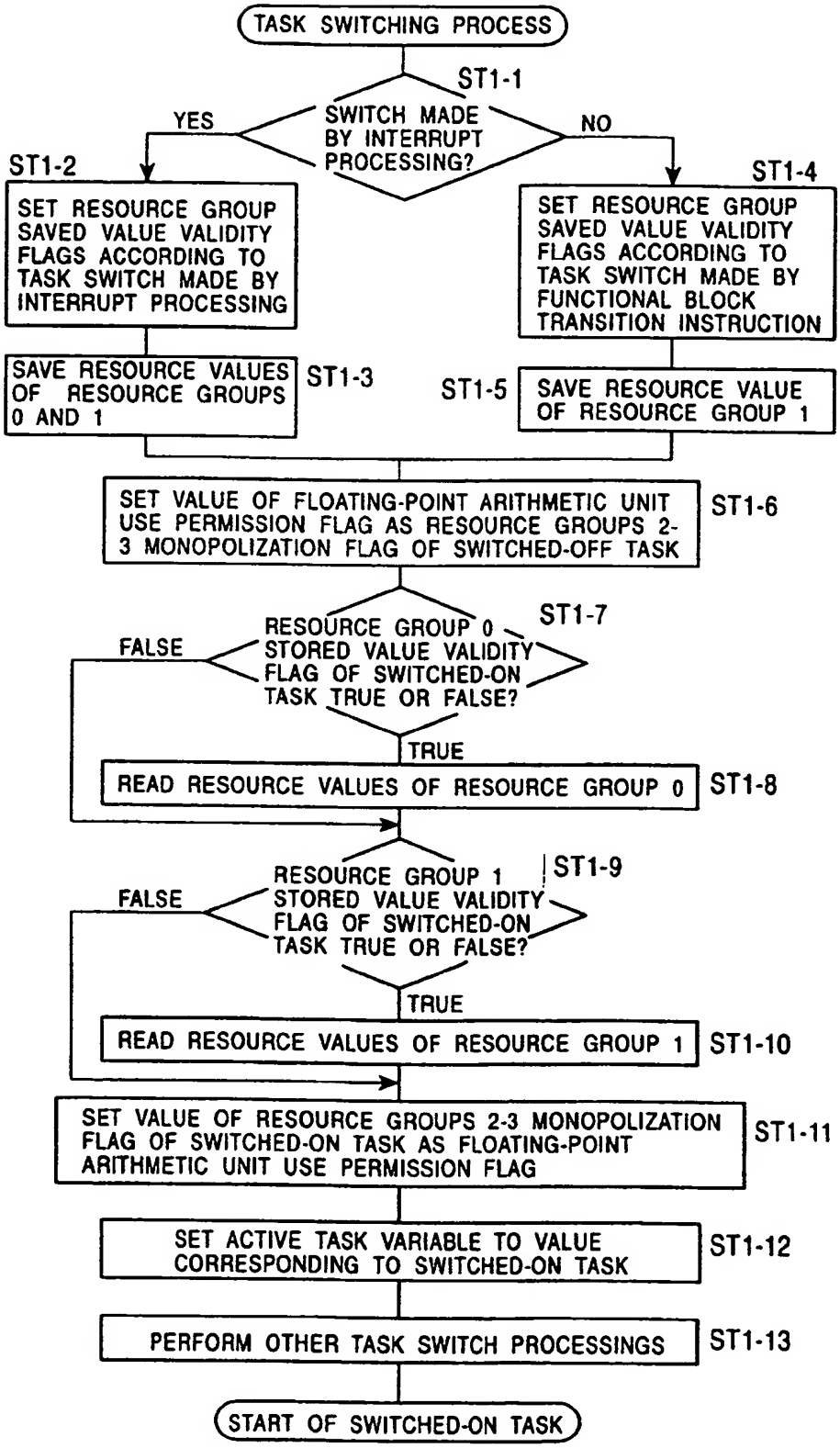

TASK INFORMATION CORRESPONDING TO TASK 0
TASK INFORMATION CORRESPONDING TO TASK 1
TASK INFORMATION CORRESPONDING TO TASK 2
TASK INFORMATION CORRESPONDING TO TASK 3
TASK INFORMATION CORRESPONDING TO TASK 4
TASK INFORMATION CORRESPONDING TO TASK 5
TASK INFORMATION CORRESPONDING TO TASK 6
TASK INFORMATION CORRESPONDING TO TASK 7

FIG. 5



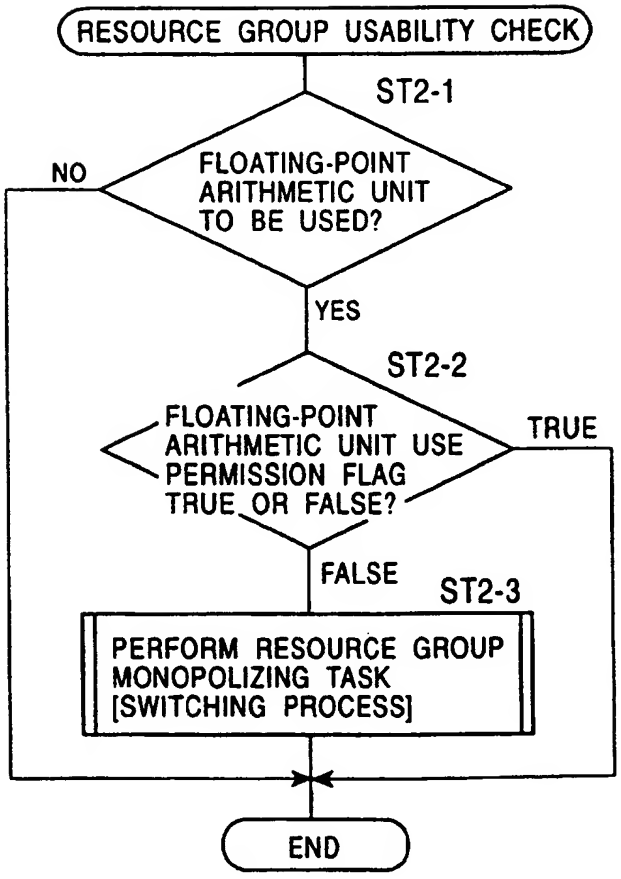
BNSDOCID: <EP 0953905A2 I >

FIG. 6



BNSDOCID: <EP 0953905A2 I >

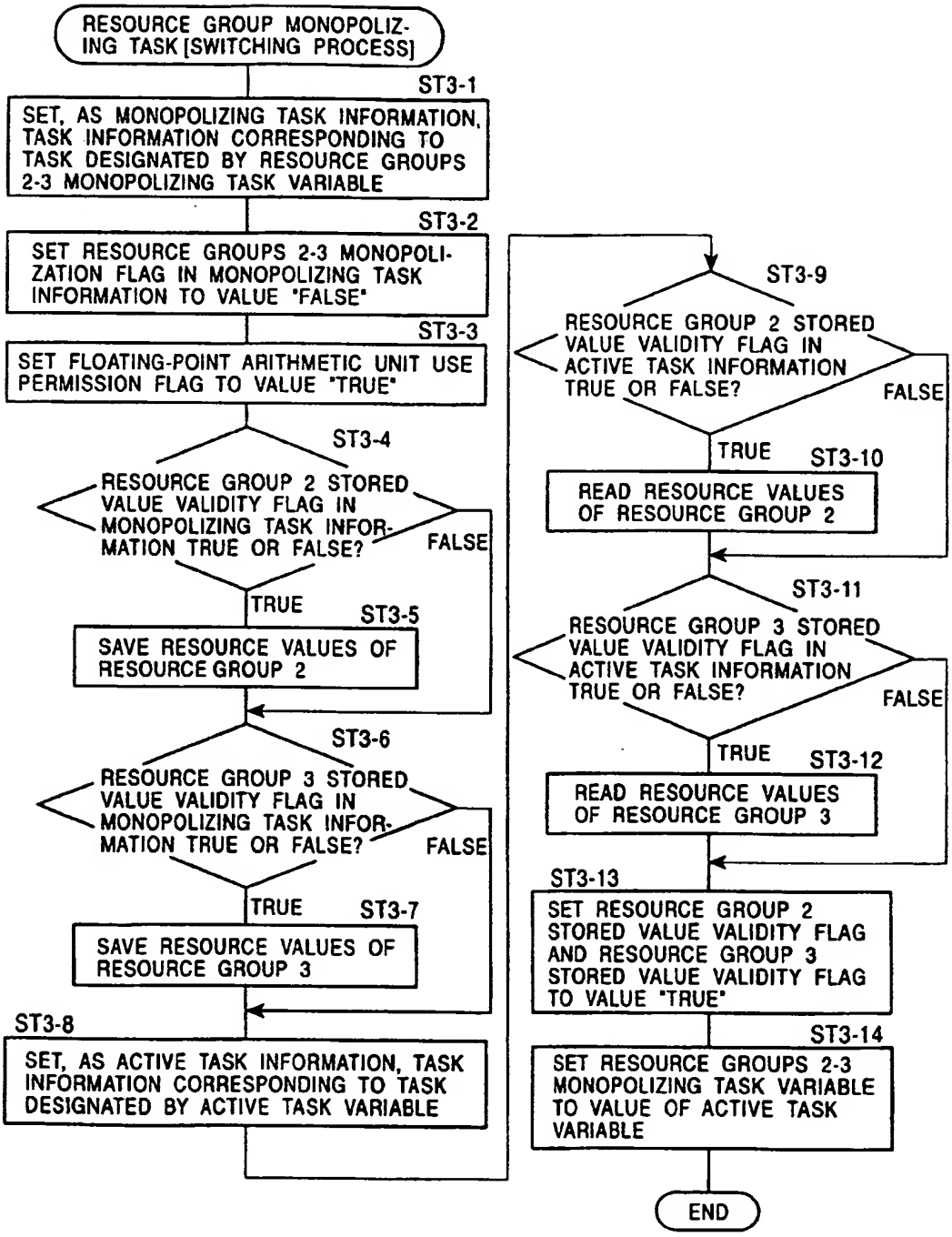
FIG. 7



BNSDOCID: <EP 0953905A2 I >

BNS page 3

FIG. 8



BNSDOCID: <EP 0953905A2 I >

FIG. 9

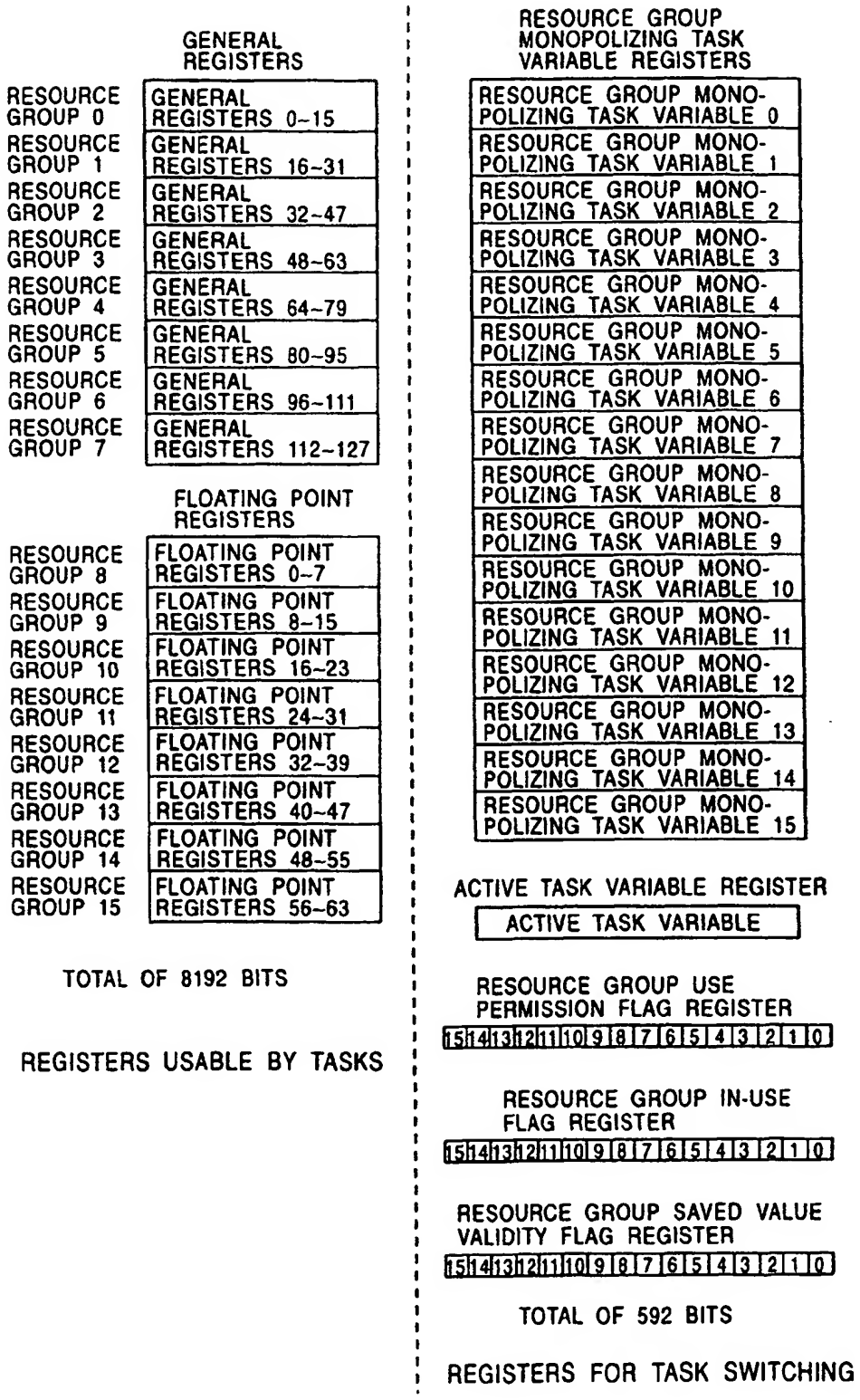
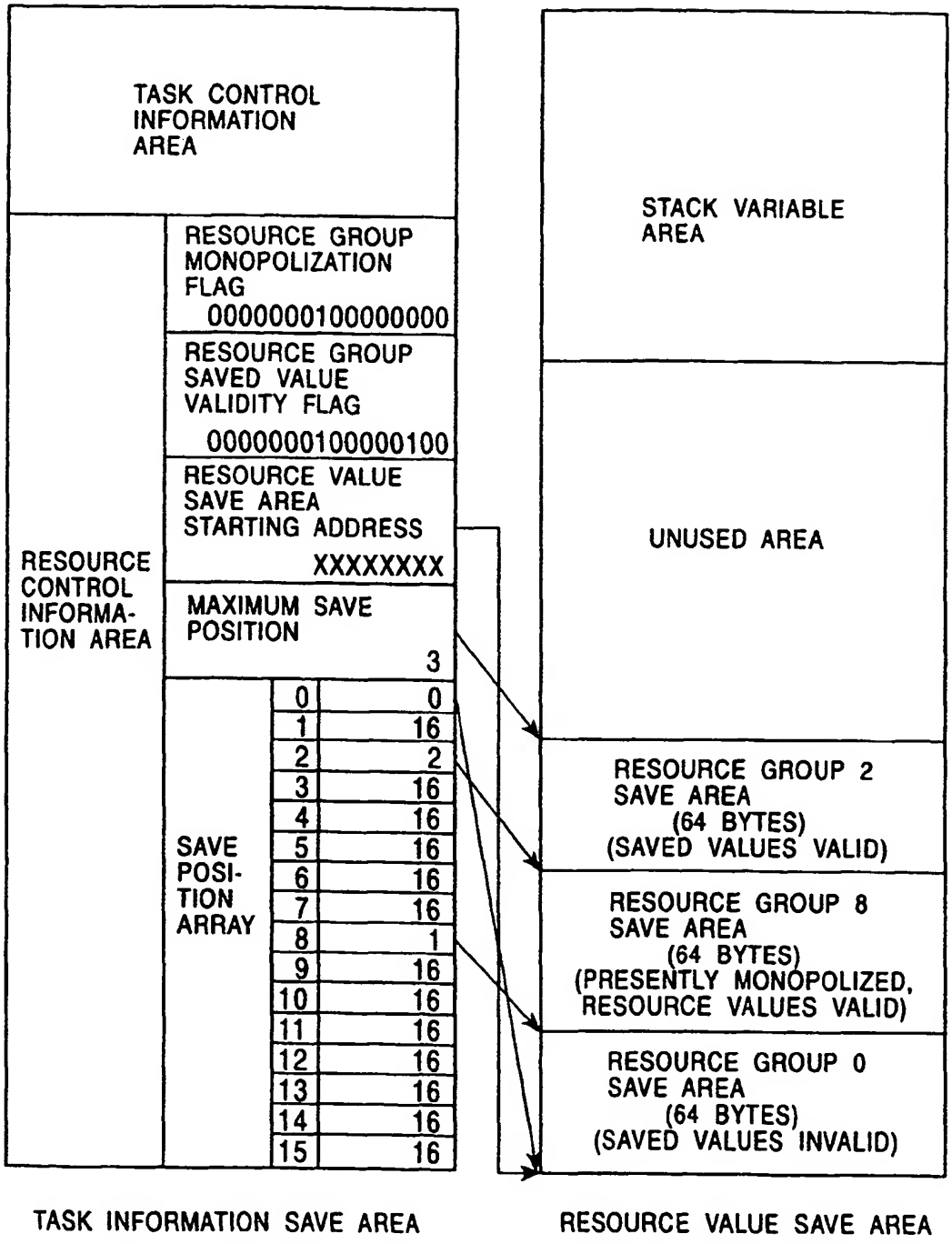
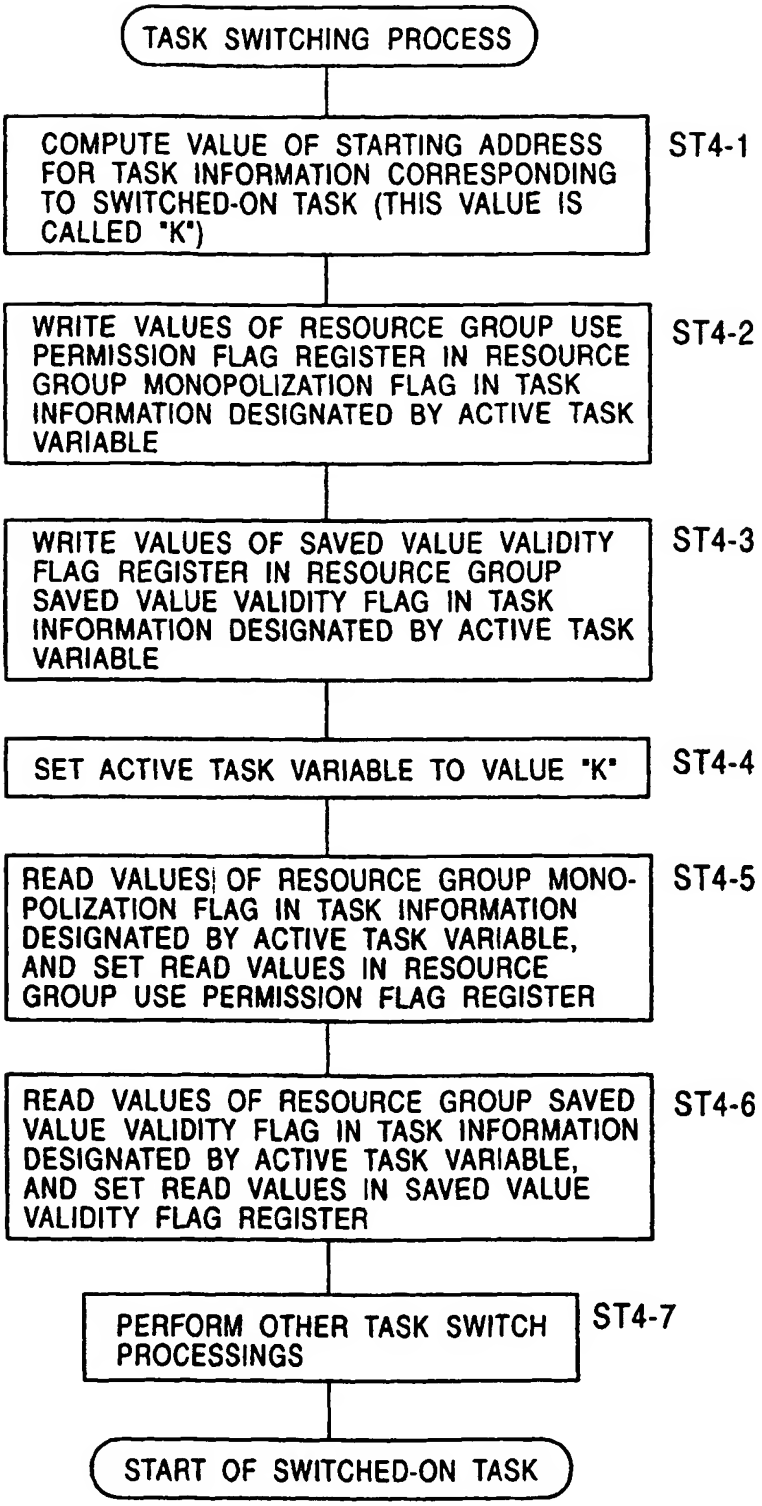

GENERAL REGISTERS
RESOURCE GROUP 0
GENERAL REGISTERS 0~15
RESOURCE GROUP 1
GENERAL REGISTERS 16~31
RESOURCE GROUP 2
GENERAL REGISTERS 32~47
RESOURCE GROUP 3
GENERAL REGISTERS 48~63
RESOURCE GROUP 4
GENERAL REGISTERS 64~79
RESOURCE GROUP 5
GENERAL REGISTERS 80~95
RESOURCE GROUP 6
GENERAL REGISTERS 96~111
RESOURCE GROUP 7
GENERAL REGISTERS 112~127
FLOATING POINT REGISTERS
RESOURCE GROUP 8
FLOATING POINT REGISTERS 0~7
RESOURCE GROUP 9
FLOATING POINT REGISTERS 8~15
RESOURCE GROUP 10
FLOATING POINT REGISTERS 16~23
RESOURCE GROUP 11
FLOATING POINT REGISTERS 24~31
RESOURCE GROUP 12
FLOATING POINT REGISTERS 32~39
RESOURCE GROUP 13
FLOATING POINT REGISTERS 40~47
RESOURCE GROUP 14
FLOATING POINT REGISTERS 48~55
RESOURCE GROUP 15
FLOATING POINT REGISTERS 56~63
TOTAL OF 8192 BITS
REGISTERS USABLE BY TASKS
RESOURCE GROUP MONOPOLIZING TASK VARIABLE REGISTERS
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 0
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 1
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 2
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 3
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 4
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 5
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 6
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 7
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 8
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 9
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 10
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 11
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 12
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 13
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 14
RESOURCE GROUP MONOPOLIZING TASK VARIABLE 15
ACTIVE TASK VARIABLE REGISTER
ACTIVE TASK VARIABLE
RESOURCE GROUP USE PERMISSION FLAG REGISTER
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESOURCE GROUP IN-USE FLAG REGISTER
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESOURCE GROUP SAVED VALUE VALIDITY FLAG REGISTER
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
TOTAL OF 592 BITS
REGISTERS FOR TASK SWITCHING

FIG.10



BNSDOCID: <EP 0953905A2 I >

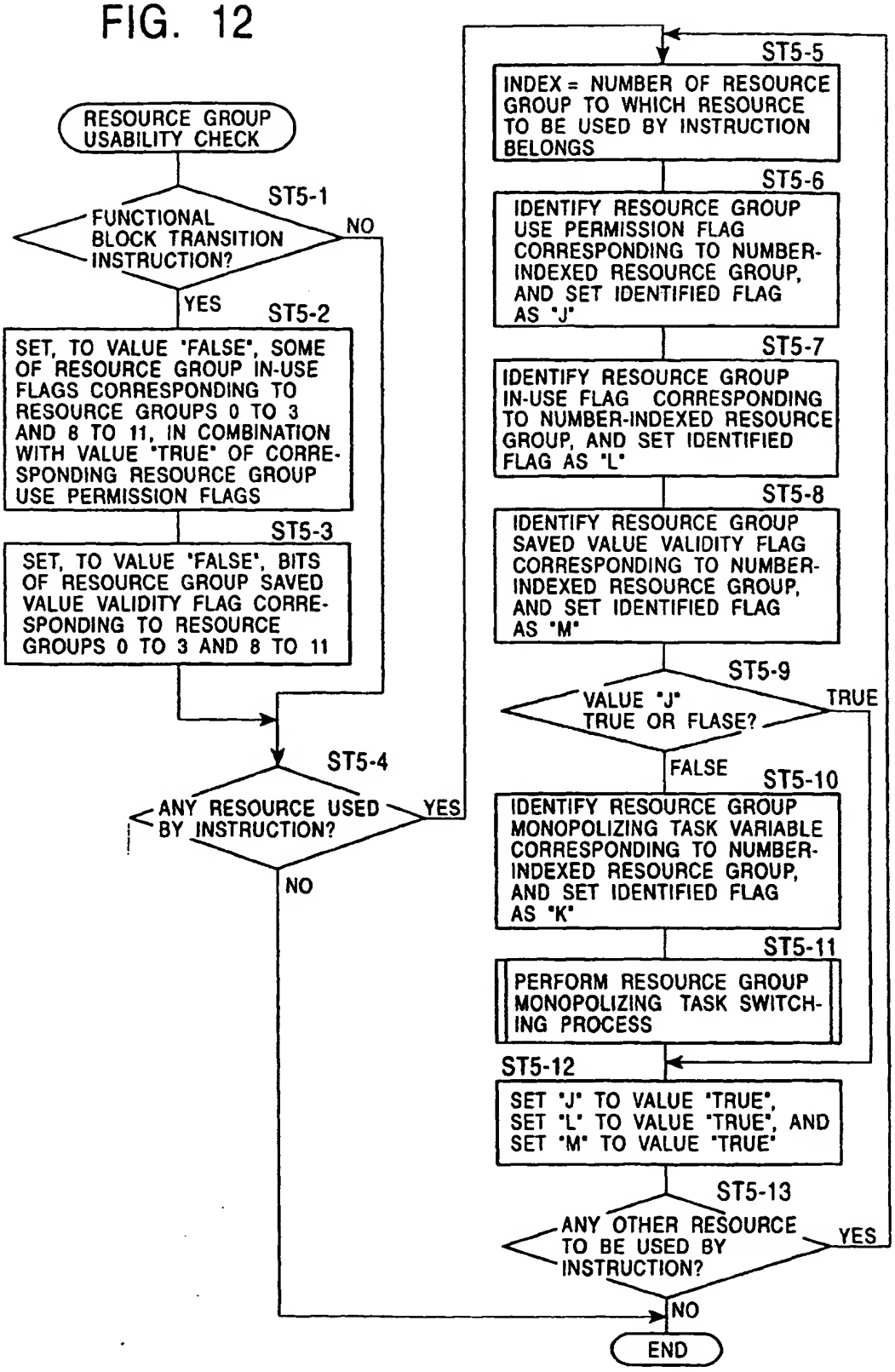
FIG. 11



BNSDOCID: <EP 0953905A2 1 >

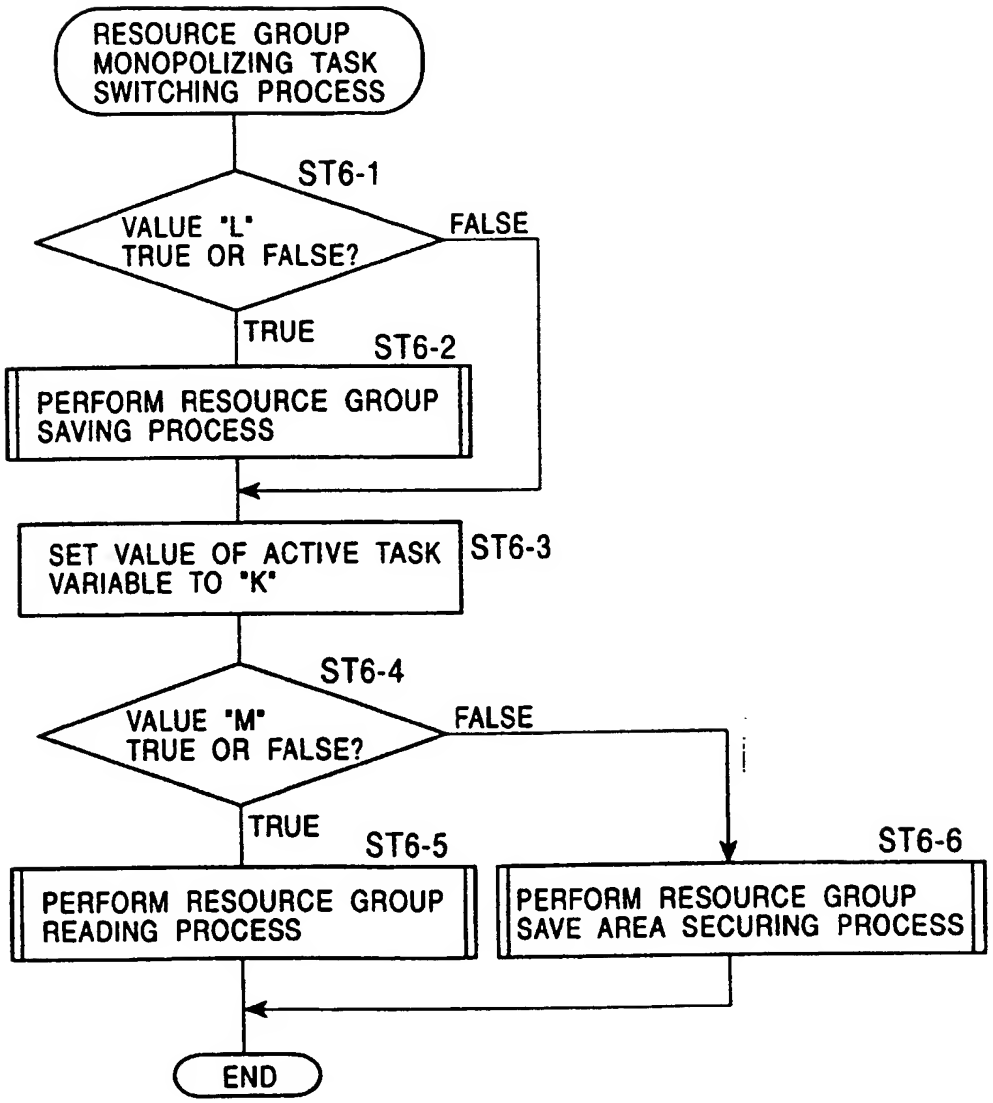
BNS page 4:

FIG. 12



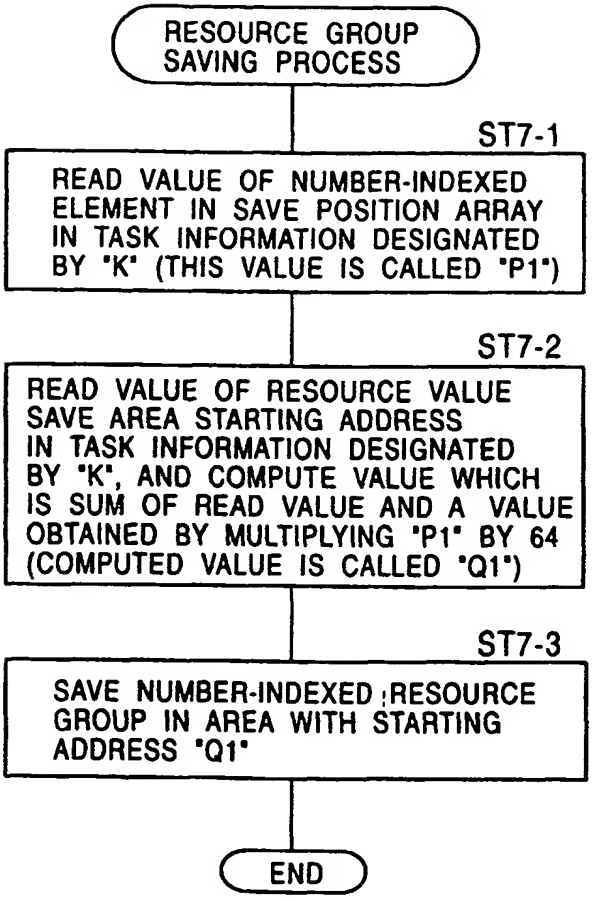
BNSDOCID: <EP 0953905A2 I >

FIG. 13



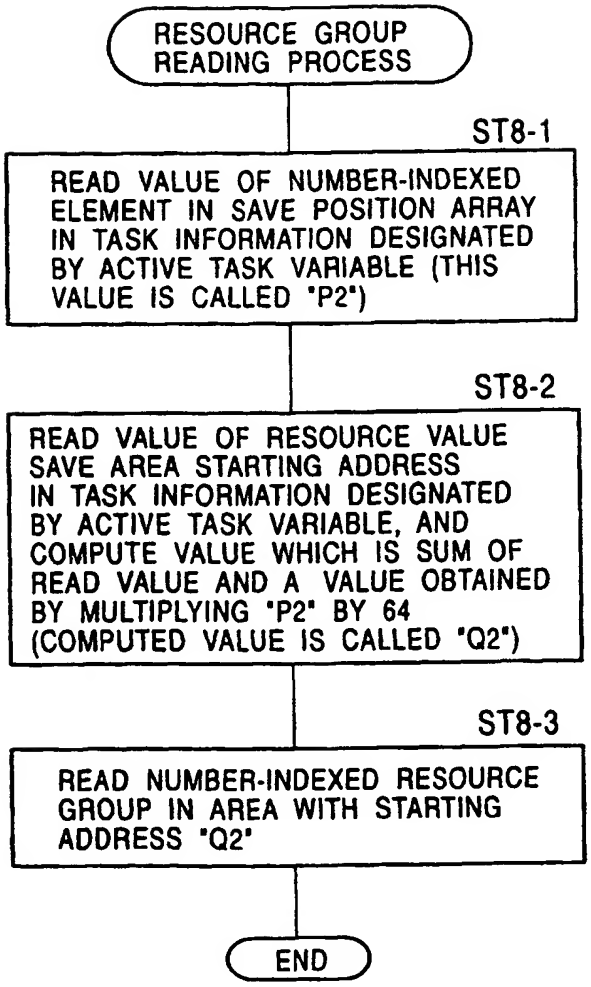
BNSDOCID: <EP 0953905A2 I >

FIG. 14



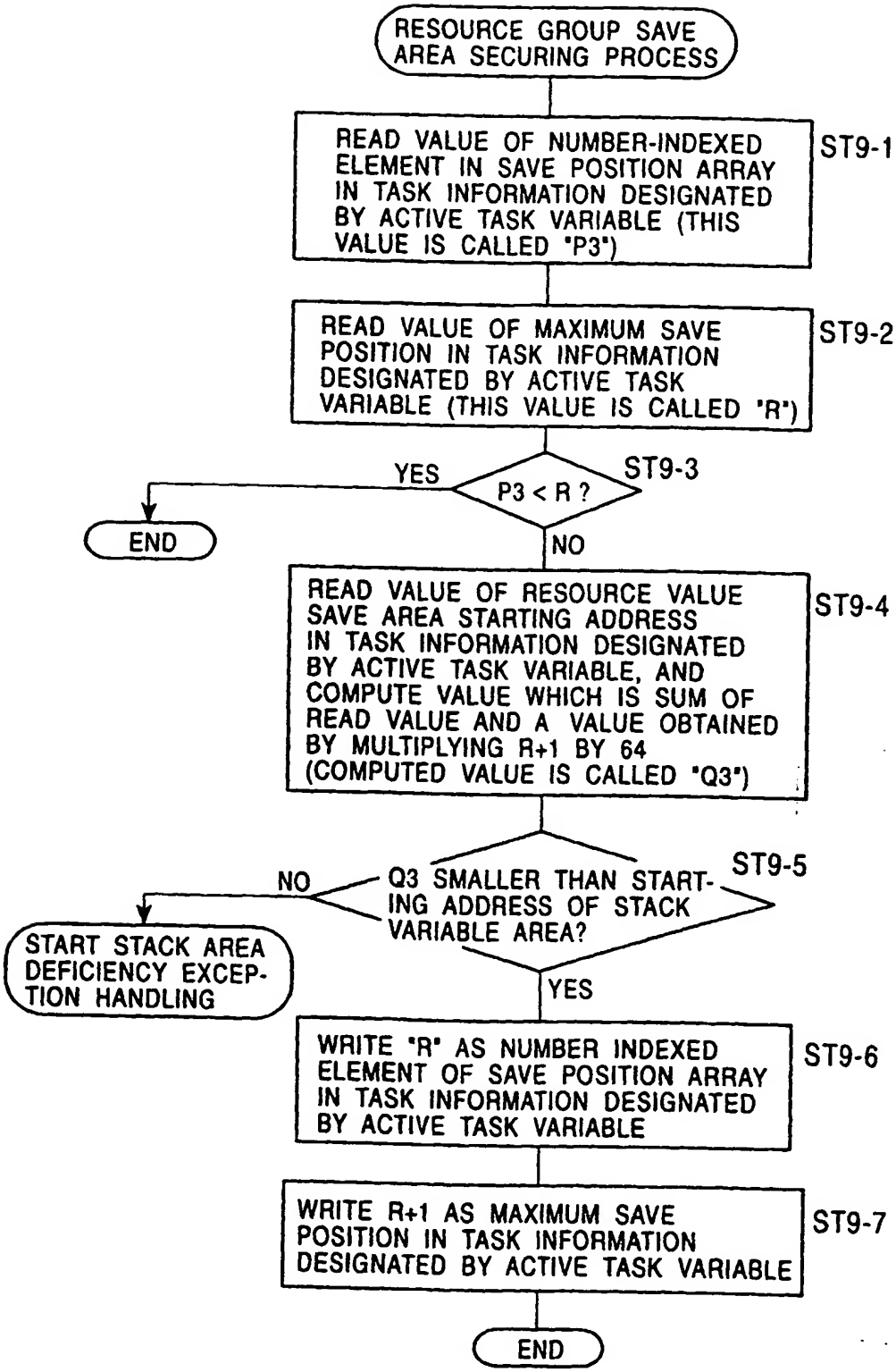
BNSDOCID: <EP 0953905A2 I >

FIG. 15



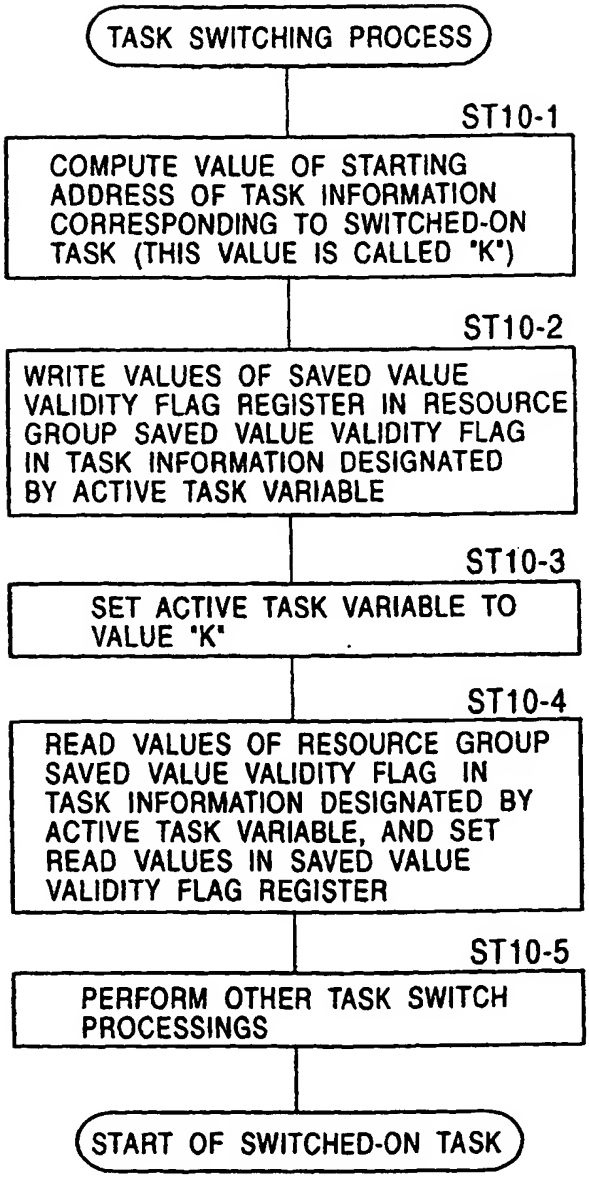
BNSDOCID: <EP 0953905A2 I >

FIG. 16



BNSDOCID: <EP 0953905A2 I >

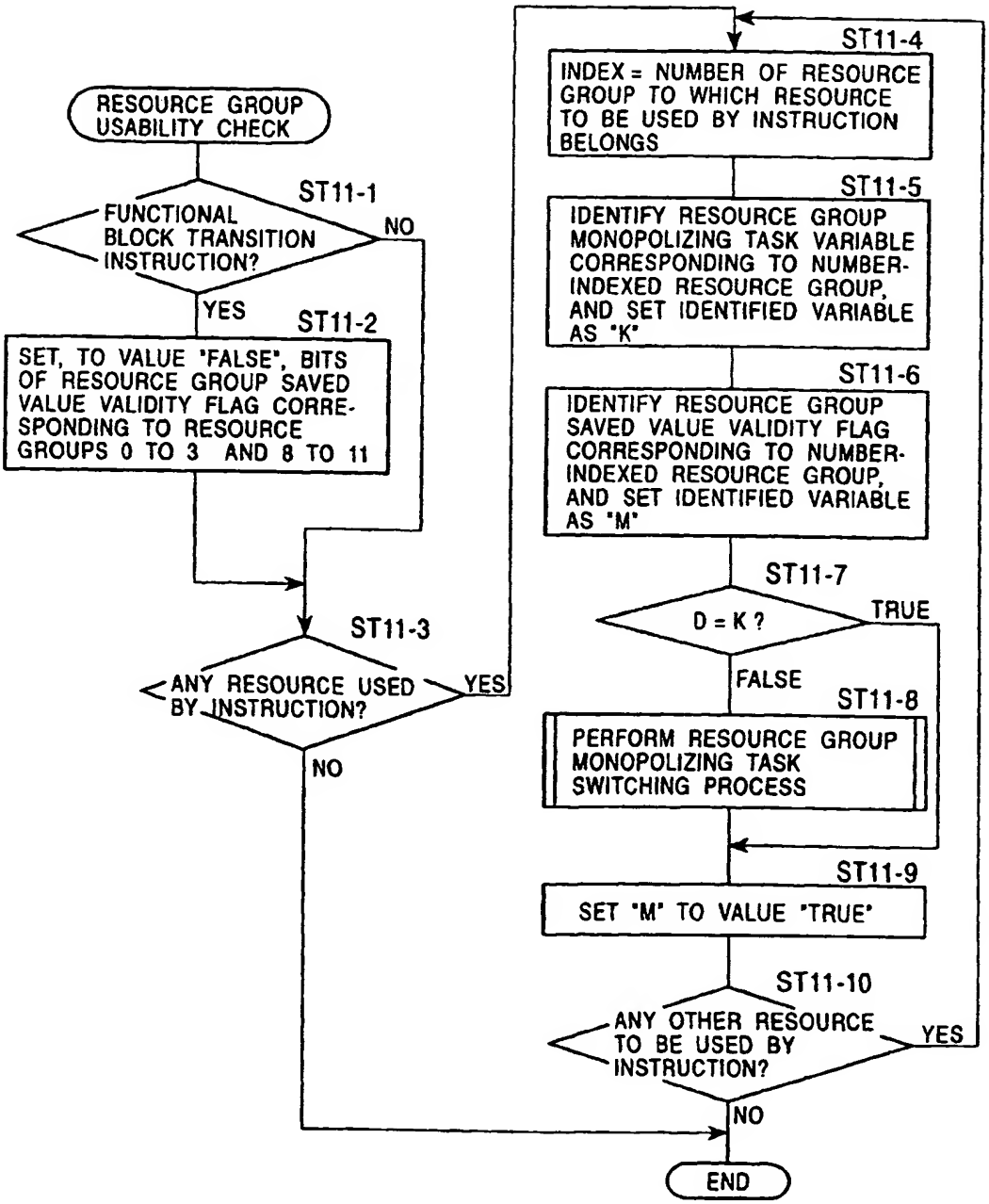
FIG. 17



BNSDOCID: <EP 0953905A2 I >

BNS page 4!

FIG. 18



BNSDOCID: <EP 0953905A2 I >

FIG. 19

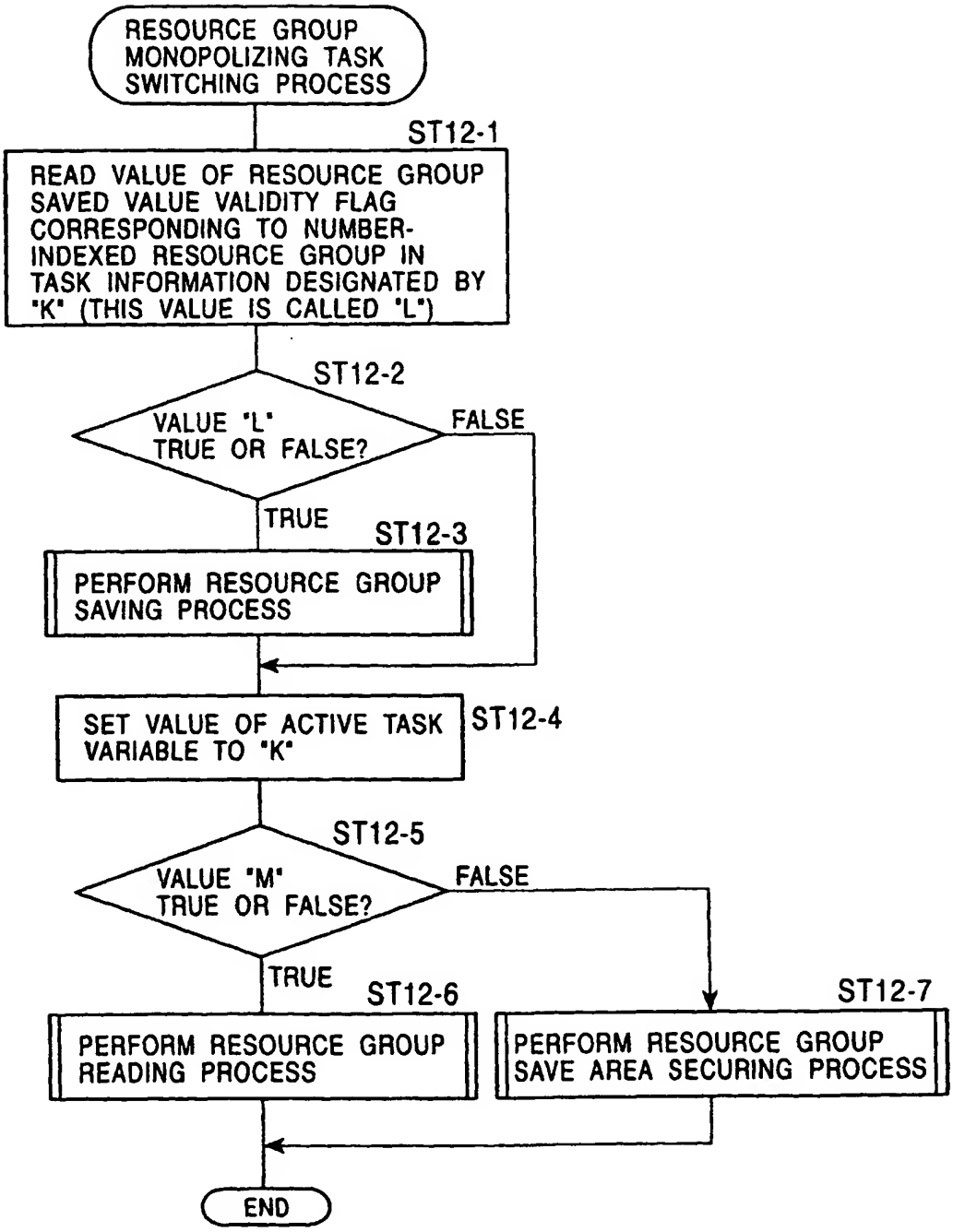
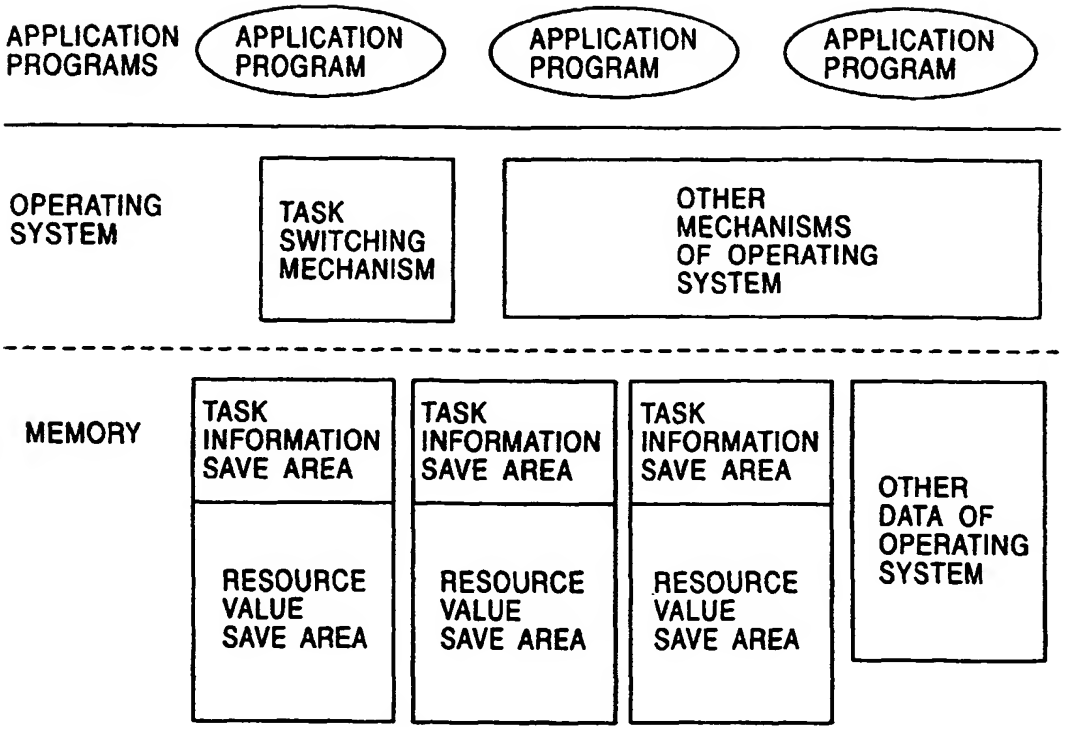

RESOURCE GROUP MONOPOLIZING TASK SWITCHING PROCESS
ST12-1
READ VALUE OF RESOURCE GROUP SAVED VALUE VALIDITY FLAG CORRESPONDING TO NUMBER-INDEXED RESOURCE GROUP IN TASK INFORMATION DESIGNATED BY "K" (THIS VALUE IS CALLED "L")
ST12-2
VALUE "L" TRUE OR FALSE?
FALSE
TRUE
ST12-3
PERFORM RESOURCE GROUP SAVING PROCESS
SET VALUE OF ACTIVE TASK VARIABLE TO "K"
ST12-4
ST12-5
VALUE "M" TRUE OR FALSE?
FALSE
TRUE
ST12-6
PERFORM RESOURCE GROUP READING PROCESS
ST12-7
PERFORM RESOURCE GROUP SAVE AREA SECURING PROCESS
END

FIG. 20



BNSDOCID: <EP 0953905A2 I >

FIG. 21A

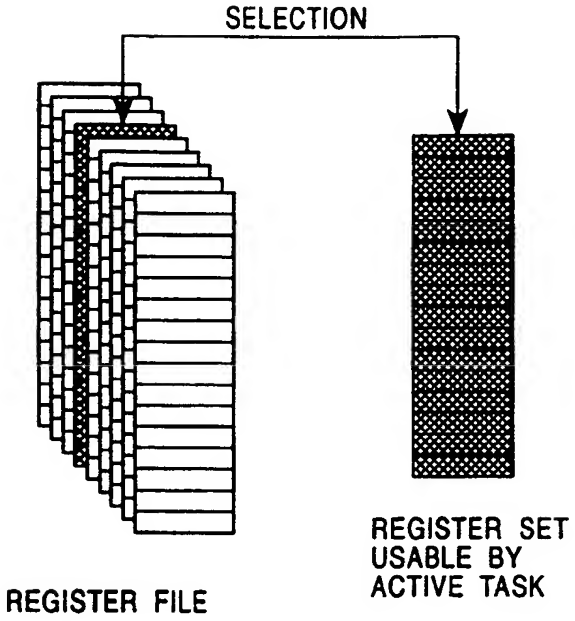


FIG. 21B

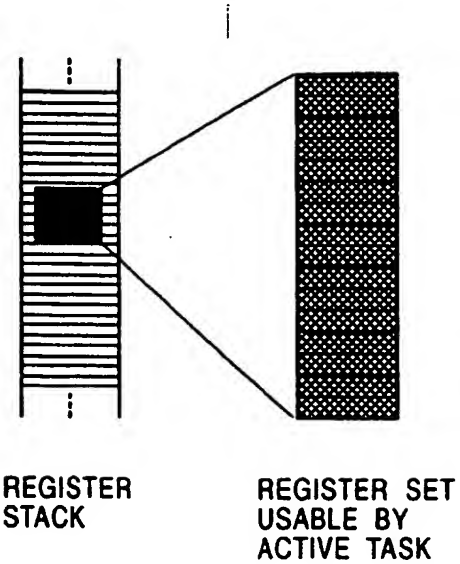
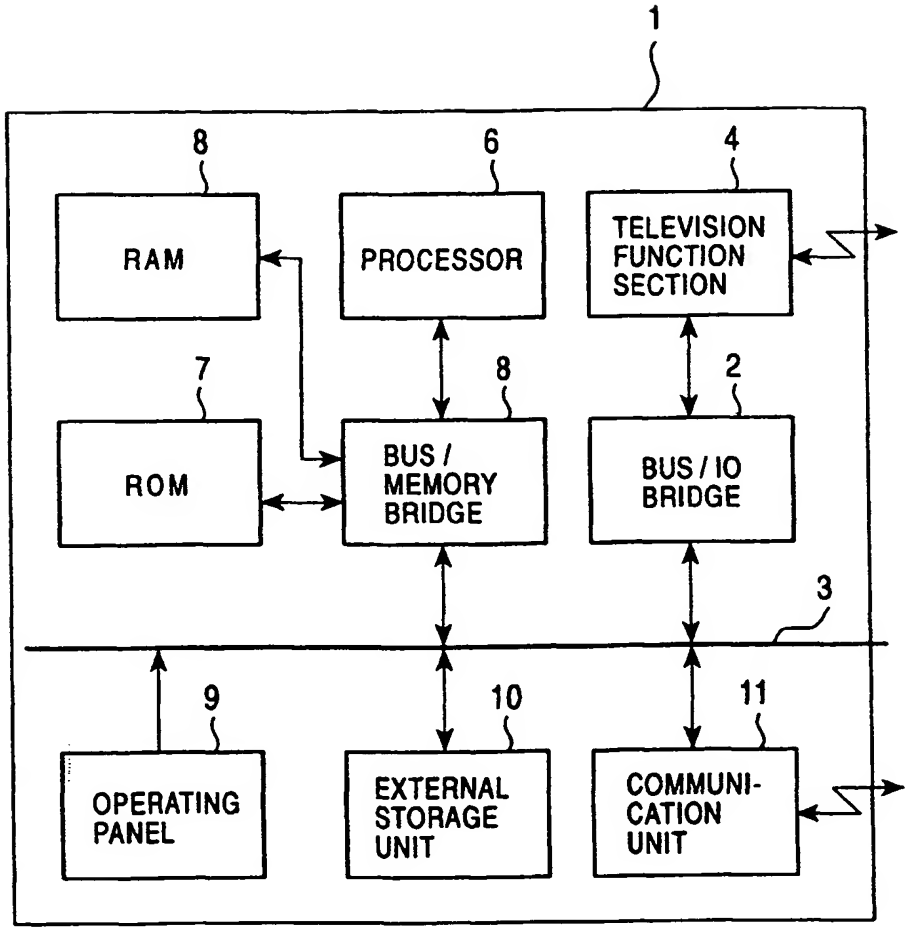


FIG. 22



BNSDOCID: <EP 0953905A2 I >